UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 790

A FEW STEPS TOWARD ARTIFICIAL 3 D VISION

Olivier D. FAUGERAS

FEVRIER 1988

A FEW STEPS TOWARD ARTIFICIAL 3D VISION

Olivier D. Faugeras
INRIA
Domaine de Voluceau
Rocquencourt, BP 105
78153 Le Chesnay
FRANCE

Abstract

This paper examines a number of important issues in Computer Vision with a strong bias toward 3D Vision. We review what we believe to be the most crucial topics in the light of what has been done at INRIA in last two years, including:

- Calibration
- Stereo
- Motion
- Shape representation
- Shape identification and location

We also identify a number of research problems to be solved in the next few years in order to make computer based visual perception a practical reality.

QUELQUES PAS VERS LA VISION ARTIFICIELLE 3D

Résumé

Nous examinons dans cet article un certain nombre de problèmes fondamentaux en Vision par Ordinateur. Nous passons en revue ce que nous pensons être les sujets cruciaux à la lumière de ce qui a été fait principalement à l'INRIA au cours des trois dernières années. Ceci inclut:

- La Calibration
- La Stéréo
- Le Mouvement
- La Représentation des Formes
- L'identification et le Positionnement des Formes

Nous identifions également dans cet article un certain nombre de sujets de recherche qu'il faudra résoudre dans les prochaines années si nous voulons que la Perception visuelle par ordinateur devienne un réalité.

1 Introduction

This paper examines a number of issues which we think should be worked on in order for 3D Vision to become available on a large scale for practical applications. For this to become true, basic scientific questions have to be answered. We need to understand what has to be computed from images [MAR82] in order to make it possible for robots to adapt to an unknown or imperfectly known environment, we need to derive robust realizations of these computations, and to implement them in such a way that real time constraints are met at reasonable costs.

We believe that this ambitious program can be achieved in our life spans, the mathematical tools are here for us to use, Computer Vision is coming of age and the hardware to make it work is also becoming increasingly available. This is not to say that all the problems have been solved but that they have been fairly well identified and that their solution is mostly a question of attacking them with the right tools and the right people. This is also not saying that they are easy problems but I do not believe that cracking them implies major theoretical breakthroughs. Solving the Vision problem is more to me like putting a man on the moon (an extremely difficult technological problem) than proving Fermat's theorem (an extremely difficult mathematical problem). This caveat is to keep all of us Vision scientists honest.

In this article, we discuss three main topics: first, how do we obtain basic 3D data from sensors, second, how do we organize these data and what do we compute with, and third, how do we use them to recognize and locate objects or places. For each of these three topics, we have tried to organize the discussion along three paths:

- what needs to be computed
- how can it be computed
- what are what we think the most challenging problems to be worked on.

2 Obtaining 3D data

There are many ways of acquiring 3D data from the environment, depending on how far we are from the objects of interest and on how accurate we want our measurements to be.

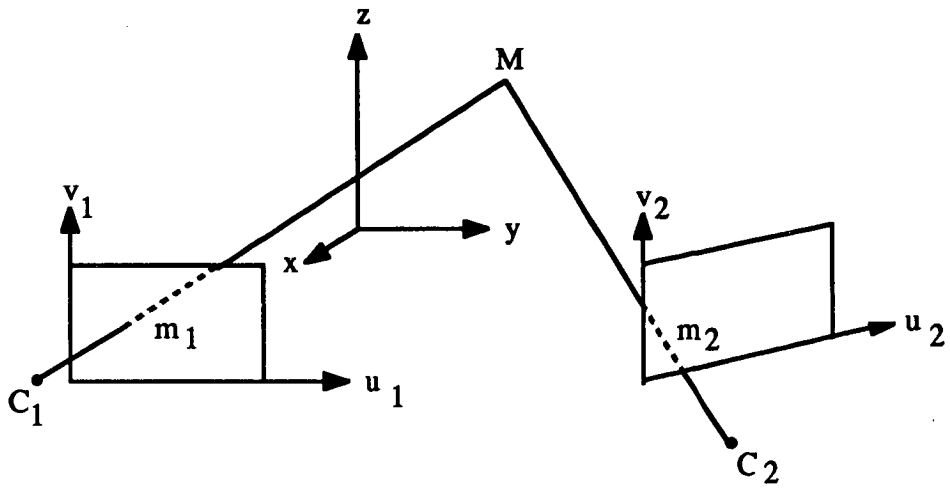


Figure 1: The problem of 3D Vision

We do not discuss here techniques such as contact sensing provided for example by tactile sensors and concentrate on non contact sensing.

2.1 Calibration

From here on, we can describe the first problem to be solved in 3D Vision as in Figure 1. Two optical systems, which we leave unspecified at the moment, yield two images $I_1(u_1, v_1)$ and $I_2(u_2, v_2)$ of the scene which is referenced to a coordinate system $(Oxyz)$. Two questions need to be answered:

- First, given a point m_1 of coordinates (u_1, v_1) in image 1, to which point m_2 in image 2 does it correspond, in the sense that they are the images of the same physical point M
- Second, given m_1 and m_2, how do we compute the coordinates of M in the coordinate system $(Oxyz)$.

2.1.1 Geometry

In order to answer these questions, we have to define models of the optical systems. The model which is universally used in the Computer Vision community is the pinhole model

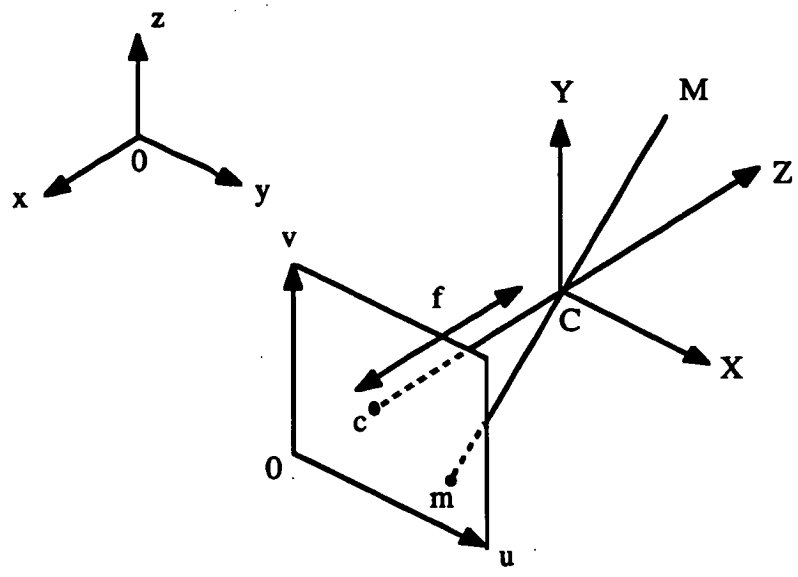


Figure 2: The pinhole camera model

shown in Figure 2. The pinhole model in its simplest form assumes that the system is defined by its optical axis perpendicular to the image plane and that the image m in that plane of a point M is formed by perspective projection with respect to the optical center C located on the optical axis at a distance f (the focal length) of the image plane. The image plane is referred to a system of coordinates (ouv) whereas space is referred to the coordinate system $(Oxyz)$. It is also convenient to consider the coordinate system $(CXYZ)$ where CZ is parallel to the optical axis and CX and CY are parallel to ou and ov, respectively. The pinhole model is then defined by its extrinsic parameters (position and orientation of $(CXYZ)$ with respect to $(Oxyz)$), and its intrinsic parameters: the coordinates in (ouv) of the point c where the optical axis pierces the image plane, and the focal length f. The problem of estimating these parameters by processing images is called the optical calibration problem and has been studied fairly widely in at least two communities:

- photogrammetry [AK71,AK74,BRO66,BRO71,FAI75], and [KAR79,KOL74,LIN72] and also [MAL71,OKA81,OKA84,WON75].
- Computer Vision [FT86,GEN79,HTMS82,IPS85,MBK81,STR84,SUT74] and [TSA85a,TSA85b,TSA86,YC78].

One basic idea is to use a number of reference points M_i of known coordinates in $(Oxyz)$, read their image coordinates and use the simple analytical relation 1 between space and image projective coordinates to estimate matrix M by least-squares techniques:

$$\begin{bmatrix} U \\ V \\ T \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

When T is different from 0, the actual image coordinates are given by $u = U/T$ and $v = V/T$. From the 3×4 matrix $\mathbf{M}$, the unknown extrinsic and intrinsic parameters can be easily computed [GAN84,FT86,TSA86].

Before we explore any further the calibration problem, let us go back for a moment to our original questions. Considering now as in Figure 3 a system of two pinhole cameras, the first question was, given a point m_1, image of an unknown 3D point M, how do we decide which point m_2 in image 2 it corresponds to. At first glance, m_2 can be anywhere in image 2, but if we look a little closer at the problem, with our simple pinhole camera model we can discover some geometric properties which will help us constrain our search. Indeed, the line C_1C_2 intersects the image planes 1 and 2 at two points E_1 and E_2 (which are either at finite or infinite distances) called the epipoles. Notice that E_1 (resp.E_2) can be considered as the "image" of C_2 (resp.C_1) by camera 1. Point M is located on the half line C_1m_1, therefore point m_2 is located on the image by camera 2 of this half line. This image is a half line starting at E_2. These purely geometrical considerations allow us to reduce considerably the search space from two to one dimension.

The second question was, given two corresponding points m_1 and m_2, how do we compute the 3D coordinates of the point M which they are the images of. At first glance, the answer seems to be the intersection of the two half lines C_1m_1 and C_2m_2. In fact, because of the uncertainties and noise, they may very well not intersect. What are then the coordinates of M ? a possible answer is to take M as the midpoint of the shortest segment between the halflines C_1m_1 and C_2m_2 (see Figure 4). We return to this problem in Section 2.7 on Motion.

At this point, from a simple pinhole camera model, we have inferred a powerful geometric constraint, the epipolar constraint, which allows us to reduce the search for corresponding points. Several new questions must then be asked:

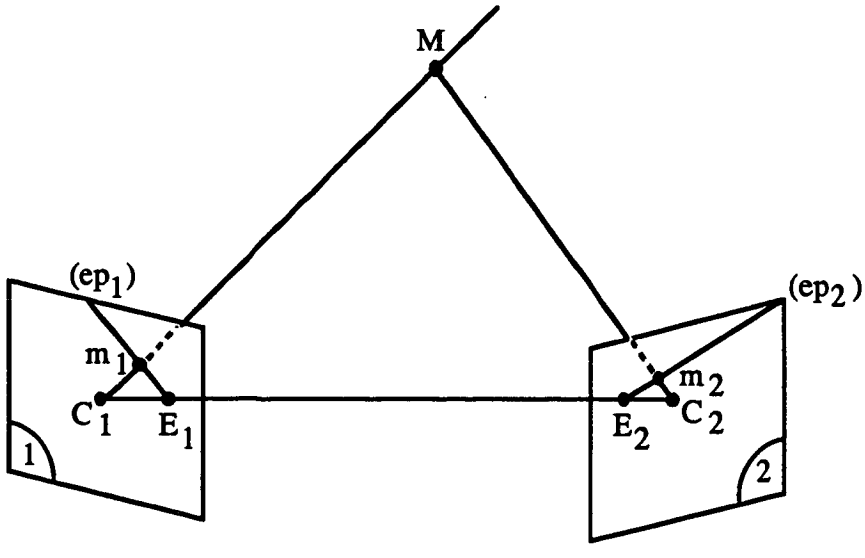


Figure 3: Epipolar Geometry

1. what is the validity of the pinhole camera model ?
2. how accurately can it be recovered by optical calibration ?
3. how can the geometric primitives which are necessary to enforce the constraints be computed from the model (i.e, for example, how do we compute the epipolar line of a point m_1 of coordinates (u_1, v_1) ?) ?

The answer to the first question is that for most of the commercially available Vidicon, *CID* or *CCD* cameras and lenses, the model is inadequate and needs to be refined in order to include effects such as lens and offcenter distortions. This implies that equation 1 does not hold anymore and nonlinearities have to be introduced. A related question is the stability in time of the model parameters. Very little is known on this subject at present. The usual solution to this nonlinearity problem is to model the real camera as the concatenation of a pinhole system followed by a nonlinear system (Figure 5). The nonlinearity is estimated and inverted and the resulting system is shown in Figure 6.

The second question is equivalent to asking how accurately do we have to measure the 3D coordinates of our reference points and their image coordinates in order to obtain a reasonable estimation of the pinhole camera model parameters. Again, very few things are known on this subject at present (but see [FT86,TSA86]).

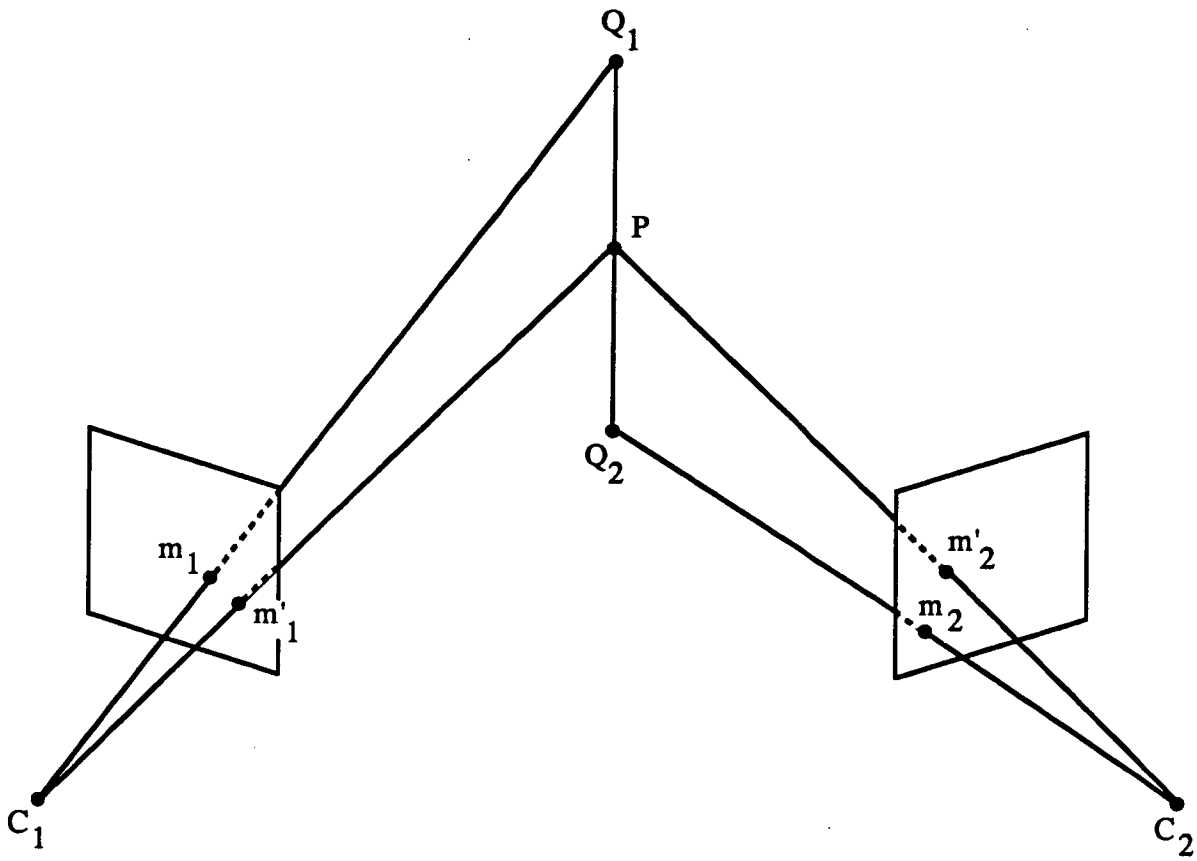


Figure 4: Reconstructing a 3D point from its two images

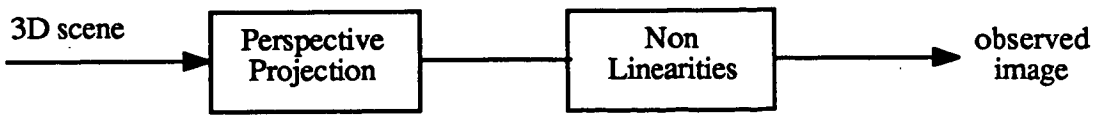


Figure 5: Realistic camera model

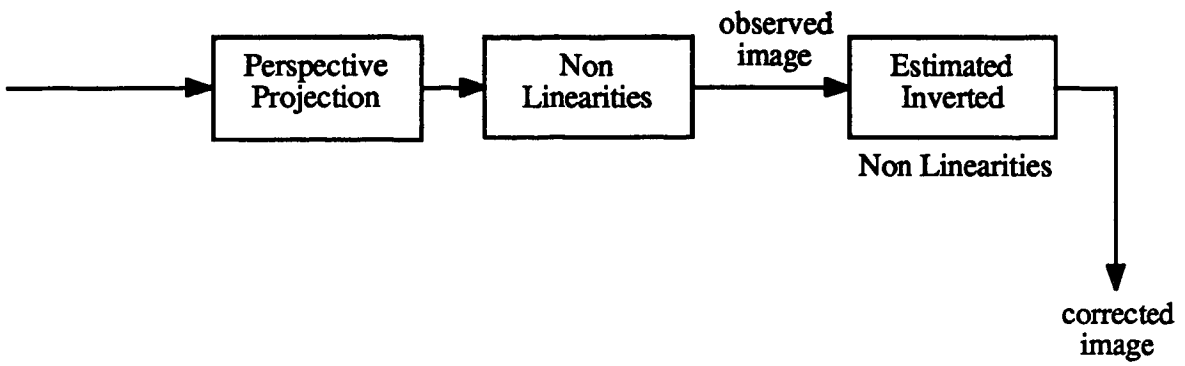


Figure 6: Pinhole camera model after eliminating the nonlinearity

2.1.2 Computing the Epipolar Geometry

Now, coming to the third question, we see that in order to answer it we need to find the coordinates of the optical centers C_1 and C_2, given the perspective matrices $\mathbf{M_1}$ and $\mathbf{M_2}$. These coordinates are obtained by solving the two systems of three linear equations in three unknowns:

$$\mathbf{M_i C_i} = \mathbf{0} \qquad i = 1, 2$$

where $\mathbf{C_i} = [x_i, y_i, z_i, 1]^T$. From there, the image coordinates of the epipoles E_1 and E_2 are computed as $\mathbf{M_1 C_2}$ and $\mathbf{M_2 C_1}$, respectively.

Let us get now to the question of computing the epipolar lines. The camera model which is usually assumed in the Computer Vision community is that of Figure 7 where the line C_1C_2 is parallel to the image rows. Both epipoles are at infinity and the epipolar lines are parallel to the scanlines, thus making the search process even simpler. In practice, it is very difficult to guarantee mechanically that this condition is satisfied, therefore the assumption is not reasonable. We must assume a general position for the cameras. In that situation, one popular approach is to rectify the images to make them appear the way they would if the camera geometry was that of Figure 7. This computationally expensive procedure called the epipolar transformation involves interpolation and accuracy is likely to be lost.

Let us study what is meant by the epipolar transformation. We have seen in the previous analysis of the epipolar geometry that in order for the epipoles to be both at infinity,

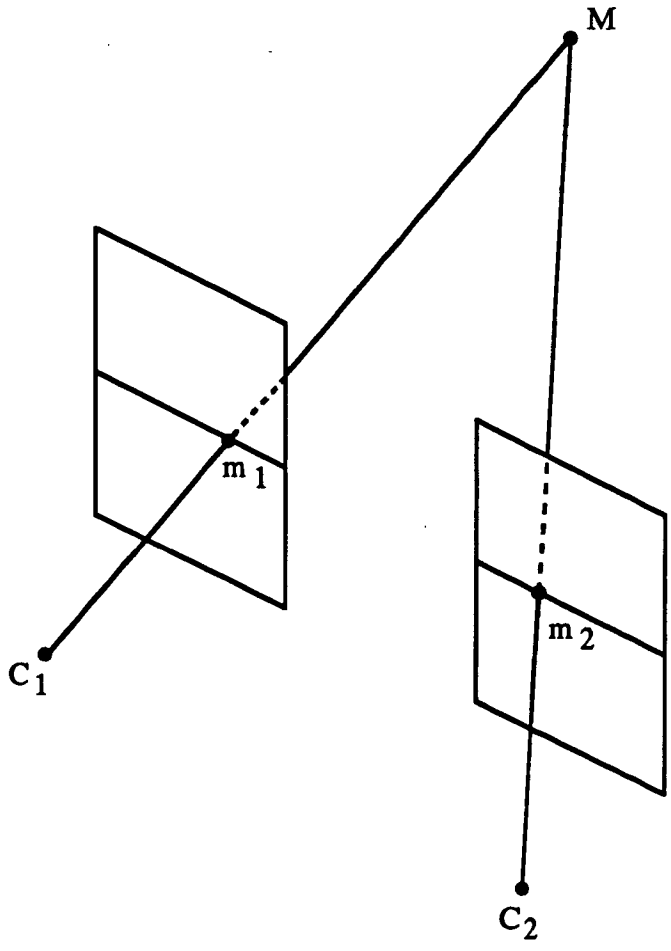


Figure 7: Simplified camera geometry: epipolar lines are image rows

the retina planes had to be parallel to the line joining the camera optical centers C_1C_2. This insures that the epipolar lines are parallel. One further condition is that they are parallel to the original scanlines. Therefore, one way to define an epipolar transformation is to project the two images onto such a plane as shown in Figure 8. Since there are many such planes, we need a criterion to choose one. Indeed, the only condition is that the plane is parallel to C_1C_2. One such possible criterion is to minimize the distorsion between the actual images and the projected ones. In the case where the two retina planes are not parallel, this probably leads to a plane parallel to C_1C_2 and to their intersection. Some work is required to understand more fully the epipolar transformation. At the moment, we believe that it is simpler to compute the equation of the epipolar line each time it is needed, especially for approaches to stereo based on symbolic matching rather than numerical correlation (see Sections 2.4, 2.5, 2.6). We show in Appendix A how this can be efficiently done.

2.2 Tokens and Features

Having discussed the problems of camera modelling, we go back to our original subject of getting 3D data. There are two main techniques for obtaining such data, stereo and structure from motion. Stereo involves a number of cameras with known positions with respect to each other, structure from motion, in the simplest case, involves one camera moving in an otherwise static environment with some unknown or imperfectly known motion. In both cases, the main question to be answered is the following:

Given a token in image 1, what is the corresponding token in image 2 ?

This matching problem has been shown to be ambiguous, and the previous question raises a number of other questions such as which tokens, features, and constraints to use to reduce this ambiguity. The discussion for tokens is common to stereo and motion. Constraints are in general different.

Tokens must be such that they can be reliably extracted from images. There are a number of possible candidates.

- The simplest one is the pixel used in the so-called intensity based correlation techniques [KMM77,GEN80,MOR80,GEN80,MOR80]. In order to identify pixels we have to attach features to them. What features should we compute in both images to help

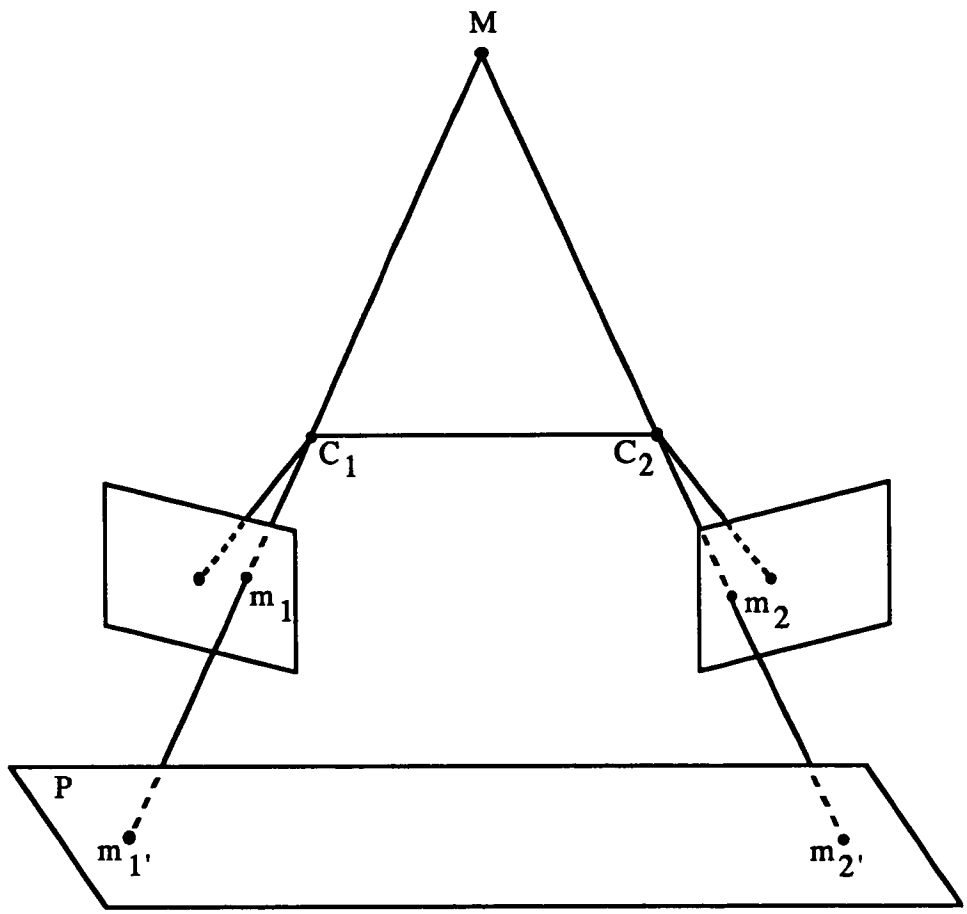


Figure 8: Epipolar transformation: plane P is parallel to C_1C_2 and therefore, the epipoles are at infinity

the correspondence problem ? one condition is certainly that if pixels m and n match, then their features should be the same or approximately the same. Since these features are computed from the measured light intensities, we have to analyse their invariance with respect to the perspective transformation.

The light reflected by the surface of an object is a function of the positions of the sources, the orientation of the surface, and the viewing direction. This behaviour can be summarized in the reflectance function [HOR74,HOR75,HOR77]. For a Lambertian surface (a completely mate surface), the reflected light is the same in all directions, therefore the intensities at two corresponding points are the same. Very few surfaces are lambertian though, the extreme case being a purely glossy one acting like a mirror for which the intensities at two corresponding points are most likely to be different. In-between these two extremes there is a whole class of surfaces for which the reflectance varies slowly with the direction of viewing and therefore for which the intensities at corresponding points are fairly close if the base line is small with respect to the viewing distance. A lot of the original techniques for computing stereo matches are based on the idea of correlating the left and right intensity images. This works well when the reflectance functions of objects satisfy the previous condition and when the disparity range is not too large to prevent large correlation distances.

- The next simplest token is the edge pixel. We can also think of grouping edge pixels together to form curves or portions of curves. The simplest curve invariant by perspective projection is the straight line. We show in Section 2.4 how to use line segments as tokens in the Stereo matching process. Segments can have both geometric and intensity based features attached to them such as length, orientation, and average constrast accross them. Care should be taken when using these features to guide the matching since they may not be invariant by perspective projection [BIN84,NP84].

  An important related point is the classification of edges. Edges in images can be classified along several dimensions. The first dimension is the shape of the intensity function at the edge: step edges are by far the most commonly used either explicitely or implicitely, but there are other types of edges such as roof, shoulder, etc $\cdots$ (see Section 3.2 and [PB85] for an application of this idea to depth images). A second dimension for classifying edges is their physical origin, i.e reflectance edges,

shadow edges, depth edges, texture edges, etc $\cdots$ If we could label every edge in the image by the class to which it belongs, then the ambiguity in matching could be considerably reduced. Indeed, edges in images are produced by physical events which fall broadly into two classes; first, markings on the surface of objects such as textures, changes in reflectance, shadows, produce edges that are invariant with respect to perspective and can therefore be reliably used for stereo matching; second, depth discontinuities may also produce edges which are invariant if the surface does not recede away smoothly. If it does, the edge in the right image will be displaced by an amount which is a function of the curvature of the surface and the viewing distance. Except for this case, edges seem to be a reliable source of information to guide the stereo matching process. Edge pixels based stereo algorithms are by far the most common [BB81,NIS84,OK85].

- Lastly, we can use image regions as tokens to be matched in the Stereoprocess. Depending on how these regions are extracted their shapes and the intensity based feature attached to them may or may not be invariant to perspective transformation [CG87].

2.3 Constraints

We have already discussed the epipolar constraint. There are others that can also be put to use, which we now examine.

2.3.1 Uniqueness

If we limit ourselves to opaque objects, one point in the first image should have at most one match in the second image. This is not true for transparent objects.

2.3.2 Continuity

The basic idea of this constraint is that the world is mostly made of objects with smooth surfaces. This means that the reconstruction function which assigns to a pair of matched points a 3D point M is smooth almost everywhere. This reconstruction function is, for historical reasons, usually summarized as a function $z = f(d)$, where z is the distance of M to the cameras and d is the so-called disparity. In order to define precisely the continuity constraint, we have to define disparity.

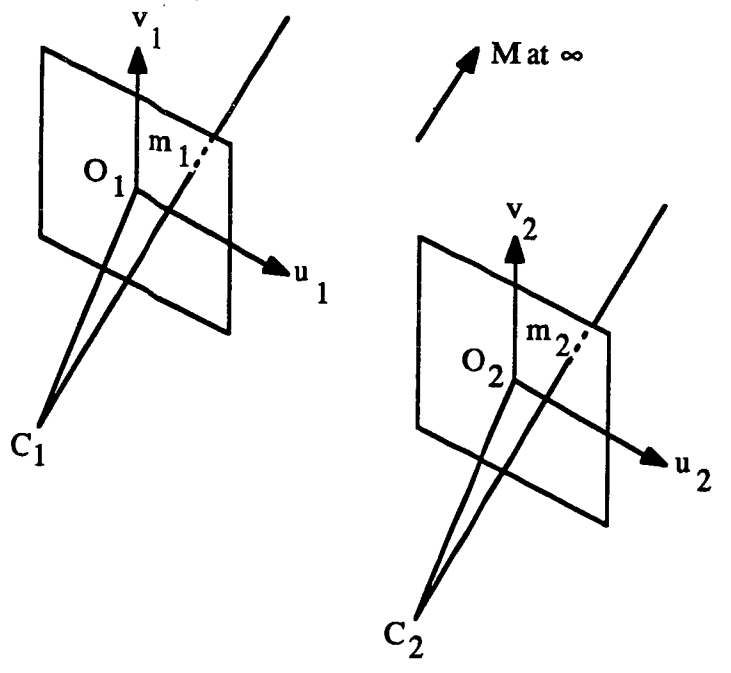


Figure 9: Simple disparity definition

Given a point m_1 of coordinates (u_1, v_1) in image 1, and its corresponding point m_2 of coordinates (u_2, v_2) in image 2, disparity is defined as $u_1 - u_2$. This definition implicitely assumes the camera geometry of Figure 9 where the two retina planes are the same. A disparity of 0 implies that the 3D point M is at infinity. If we bring point M to m_1 along the half infinite line $C_1 m_1$, the disparity will decrease from 0 to $-d_{12}$, where d_{12} is the distance between C_1 and C_2. There is also a simple relationship between disparity and the distance of the 3D point M, distance measured from the two camera planes (Figure 10):

$$d = u_1 - u_2 = d_{12} f / z$$

Fronto-parallel planes are locus of points with constant disparity.

When we assume the general camera position of Figure 11, there is not anymore a simple relationship between the coordinates (u_1, v_1) and (u_2, v_2) of the two corresponding points when M is at infinity. To find out what happens, let us draw the Figure in the epipolar plane $C_1 C_2 m_1 m_2$ (Figure 12). When M travels from infinity to m_1 along the line $C_1 m_1$, m_2 varies from $m_{2\infty}$ to m_{20} (m_{20} is the point of intersection of line $C_2 m_1$ and the epipolar line $E_2 m_{2\infty}$). What is important for the matching process, is that the point corresponding to m_1 is to be found on the epipolar segment $m_{20} m_{2\infty}$.

Nonetheless, it is possible to define a notion of disparity for a point M with the help

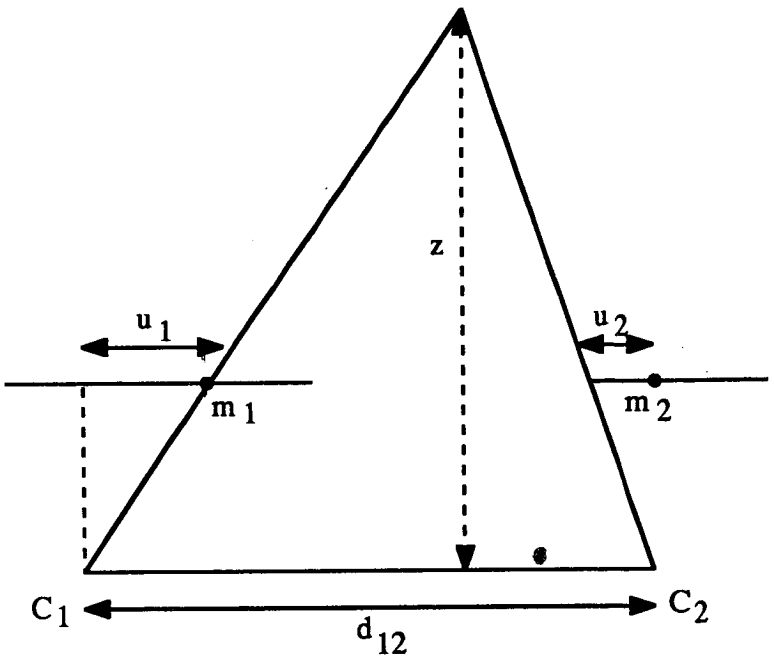


Figure 10: Relation between depth and disparity

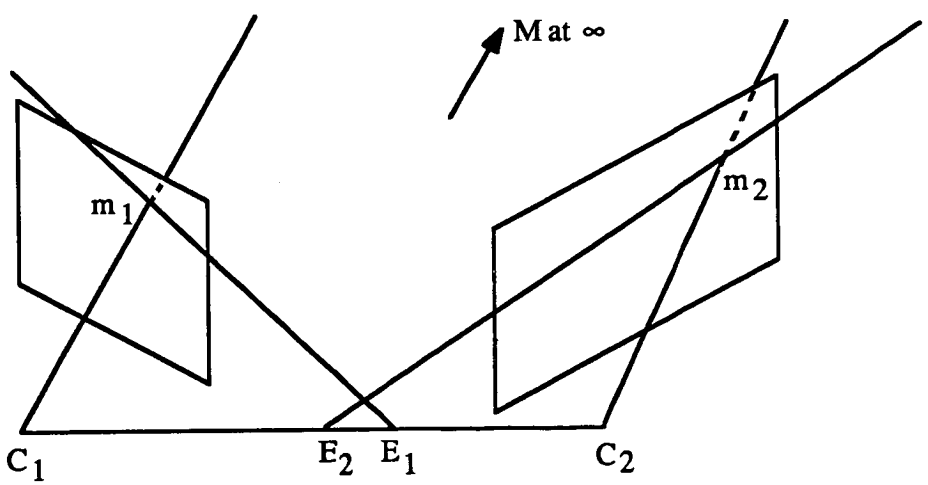


Figure 11: General camera position

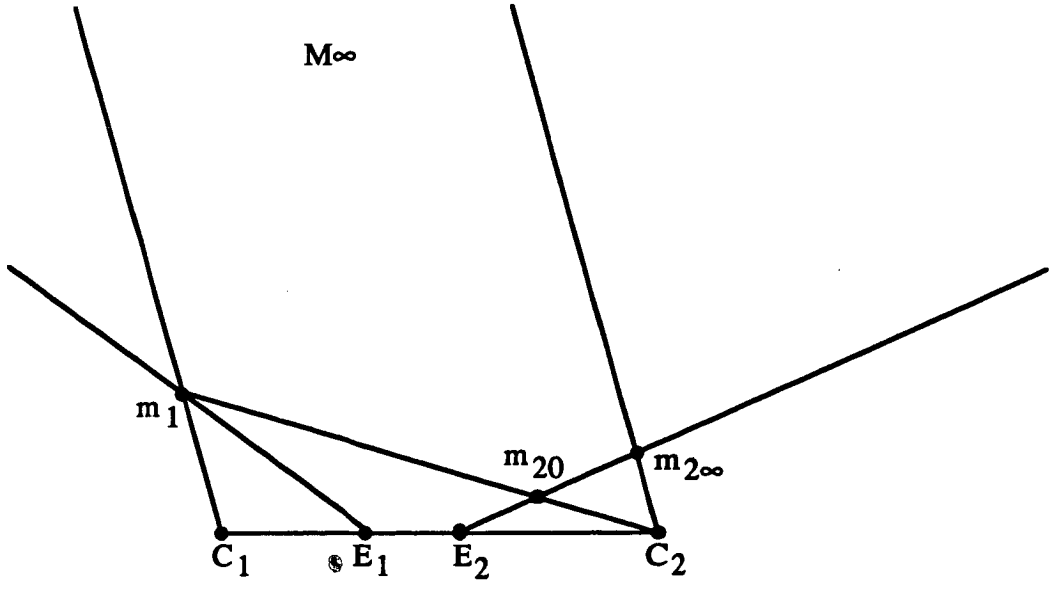


Figure 12: Epipolar plane configuration

of Figure 13. Indeed, there is a known relationship between the abscissas u_1 and u_2 of m_1 and m_2 along the epipolar lines, and the coordinates (x, z) of M where z is the distance to the axis C_1C_2 (this relation is the 2D analog of equation 1):

$$u_1 = (a_1x + b_1z + c_1)/(d_1x + e_1z + f_1)$$

Similarly:

$$u_2 = (a_2x + b_2z + c_2)/(d_2x + e_2z + f_2)$$

Where the coefficients $a_i, b_i, c_i, d_i, e_i, f_i, i = 1, 2$, depend only on the geometry of the two epipolar lines and the cameras centers. Eliminating x between these two equations yields:

$$z = (c_2 - u_2f_2)(u_1d_1 - a_1) - (c_1 - u_1f_1)(u_2d_2 - a_2)/(b_1 - u_1e_1)(u_2d_2 - a_2) - (b_2 - u_2e_2)(u_1d_1 - a_1)$$

Disparity can then be defined as $d = 1/z$.The origins on the u_1 and u_2 axis can be chosen at the epipoles E_1 and E_2, respectively. Constant disparity now means constant z. Points of equal disparities are on circular cylinders of axis C_1C_2.

If we assume that the objects that we are looking at are smooth almost everywhere, then the disparity constraint can be introduced in the following manner. Let M be a point in 3D space with projections m_1 and m_2 on retinas 1 and 2 with a disparity d as previously defined. Then, a neighbor n_1 of m_1 in retina 1 should find a match n_2 in retina 2 with a disparity close to d.

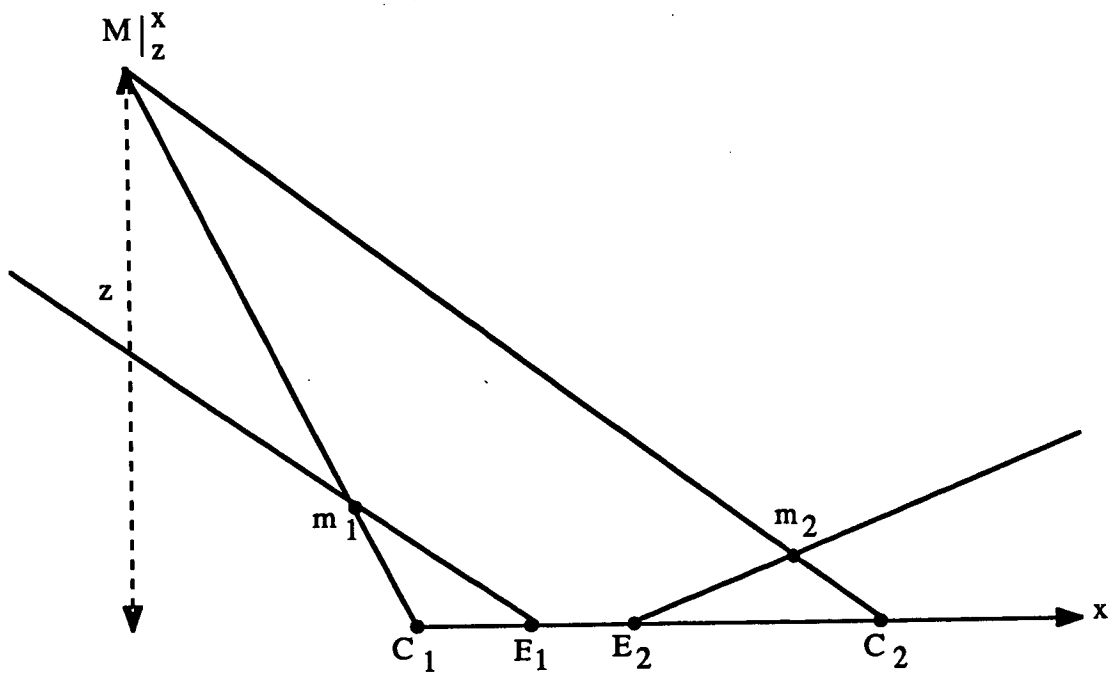


Figure 13: General definition of disparity

2.3.3 Ordering

Let us consider a 3D point M_1 and its projections m_1 and n_1 in retinas 1 and 2, respectively. Figure 14 shows the corresponding epipolar plane configuration. Now let us choose a point M_2 in the cone defined by M_1, C_1, C_2 containing the base line C_1C_2 (cross-hatched in Figure 14). M_2 has images m_2 and n_2 in retinas 1 and 2, respectively. Let us now look at the order of the images along the epipolar lines: we have (E_1, m_2, m_1) for retina 1 and (E_2, n_2, n_1) for retina 2. In other words, the images of M_1 and M_2 appear in the same order along the epipolar lines when M_2 is in the previous cone. It is easy to see that the converse is true when M_2 varies in the other cone defined by M_1, C_1, and C_2 (the one which is not cross-hatched in Figure 14).

There are several arguments for calling the cross-hatched zone in Figure 14 the forbidden zone attached to M_1:

1. If the distance of M_1 to the base line C_1C_2 is large with respect to the length of C_1C_2, then the angle $C_1M_1C_2$ is small and the probability for a point M_2 to fall into it is quite small but non zero.

2. If we assume that M_1 and M_2 are situated on an opaque object of non zero thickness as shown in Figure 15, then M_1 and M_2 cannot possibly be seen simultaneously by

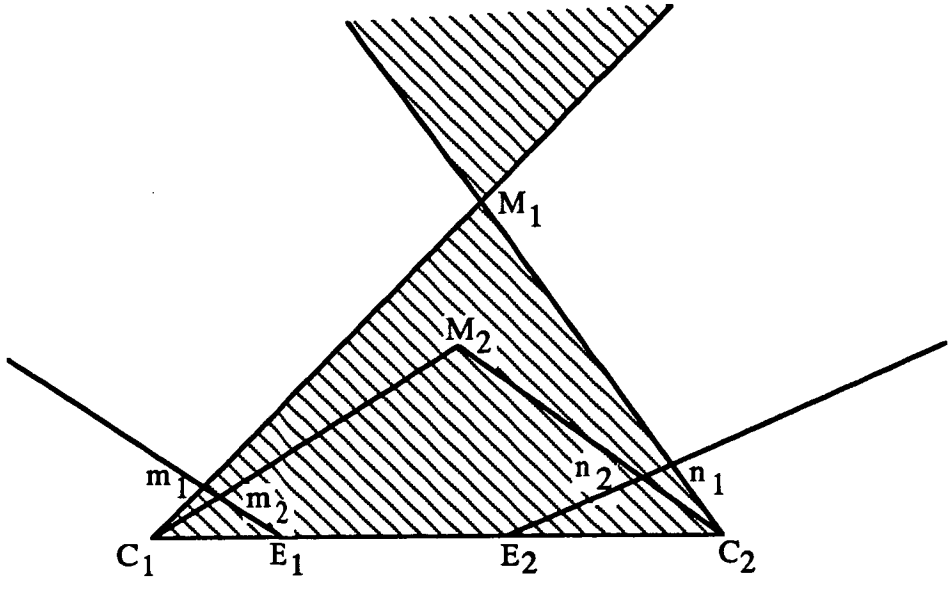


Figure 14: Forbidden zone attached to M

retinas 1 and 2.

Since the fact that M_2 is in the forbidden zone defined by M_1 can be easily checked by checking the ordering of their images along the epipolar lines, this constraint can be used to eliminate matches for m_2, given the match (m_1, n_1).

In practice, it is difficult to eliminate the whole cross-hatched cone of Figure 15, because of configurations such as the one depicted in Figure 16. In this Figure, the points M_1 and M_2 are seen to belong to two different objects for which the ordering constraint does not apply. It seems therefore more reasonable to force only the neighbors of M_1 within a small neighborhood (for example a disc of radius ε) to belong to the non forbidden zone (Figure 17).

2.3.4 Geometric constraints

In some applications, we may know a priori that the surfaces of objects have locally some known analytic form. The simplest form of this assumption is to assume that objects are locally/piecewise planar, i.e are well approximated by their tangent planes almost everywhere, that is except at discontinuities. We show in Section 2.5 how to use this extra constraint.

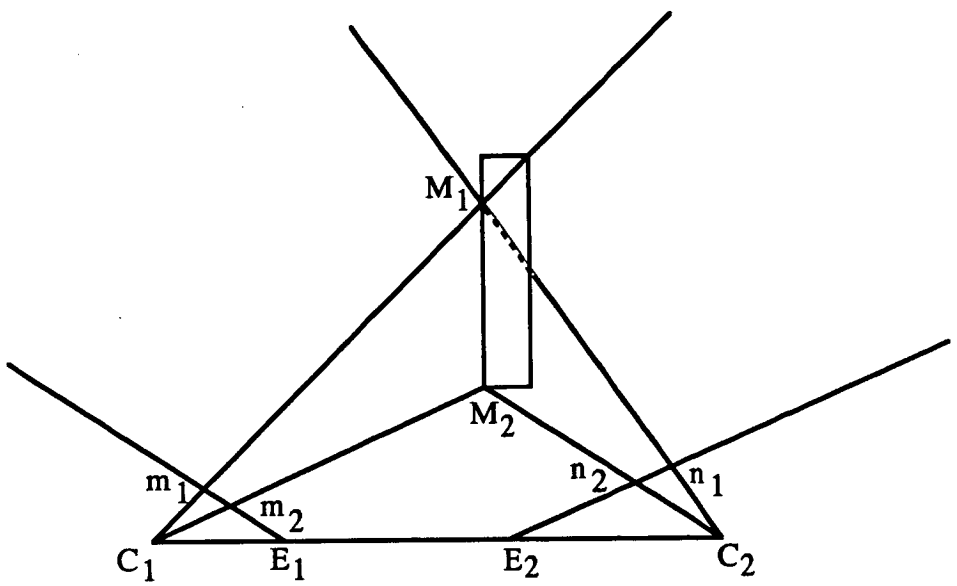


Figure 15: M_1 and M_2 are not both visible from retina 2

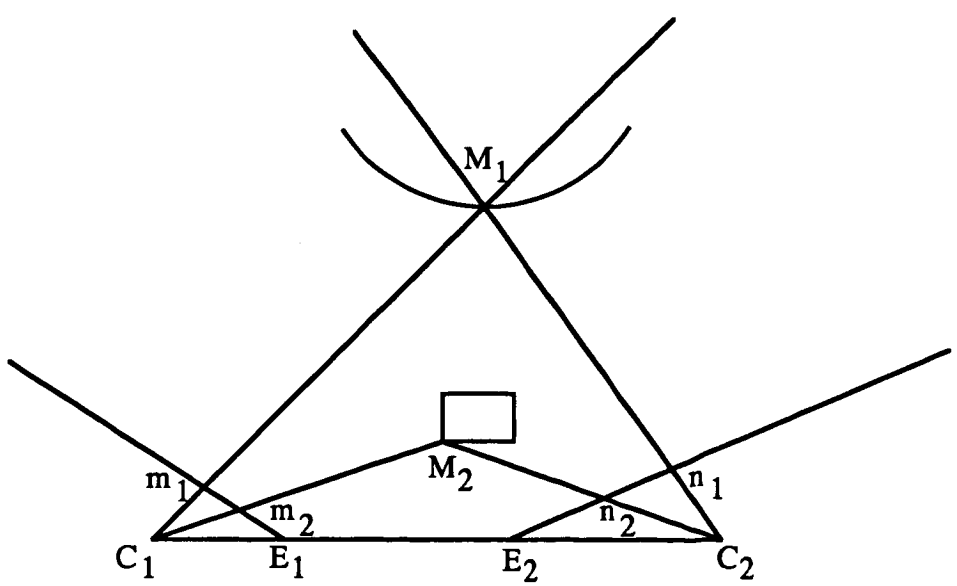


Figure 16: Breaking the ordering constraint

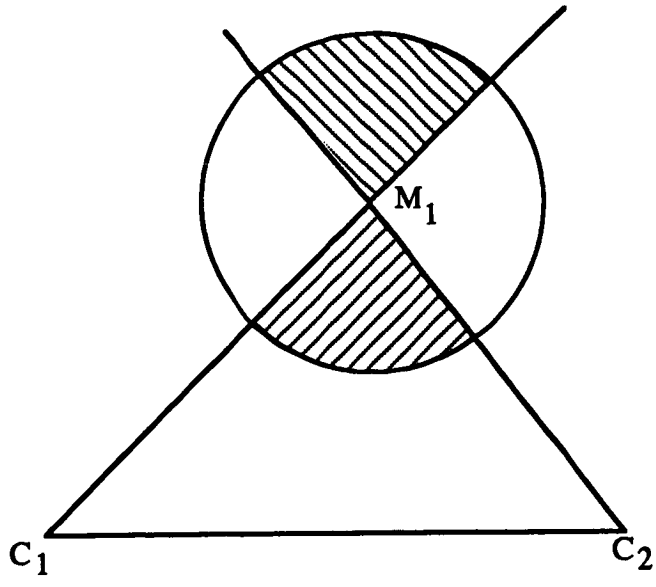


Figure 17: Actual forbidden zone

2.4 Finding the Stereo matches

There has been a large number of algorithms published to solve the stereo matching problem and we do not intend to review all of them. The interested reader is referred to the following references [MP76,MP79,BB81,GRI81,OK85]. We want to discuss here an algorithm developed at INRIA by Ayache and Faverjon [AF85] and Ayache and Lustman [AL87] because it is a good example of the application of a powerful technique, Hypothesis Prediction and Verification, to the matching of symbolic primitives, technique which we have used successfully in a number of other applications (see Section 4).

The symbolic primitives are line segments described by a number of features. The algorithm uses four constraints to reduce the size of the search space:

- The epipolar constraint, suitably modified for the case of line segments.
- The geometric similarity constraint to forbid matches between dissimilar segments.
- The continuity constraint implemented as a disparity gradient constraint.
- The uniqueness constraint.

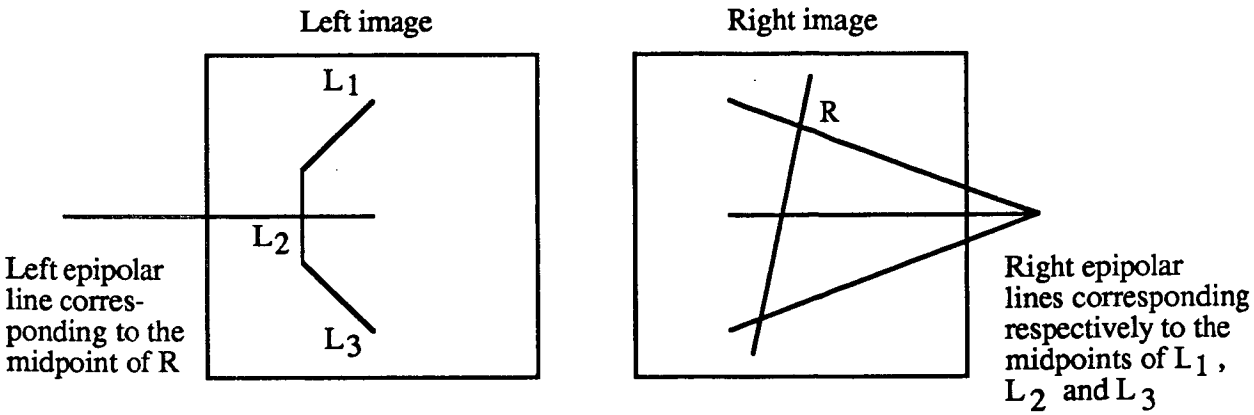


Figure 18: Multiple segment matches

2.4.1 Building the symbolic descriptions

Both images are represented as sets of line segments. These segments are extracted by first finding edge points, and then approximating those points by straight line segments. Each segment is represented by a number of features, both geometric (coordinates of midpoint, length, orientation) and intensity based (contrast across the segment, average intensity of image gradient along the segment, etc ...). A neighborhood structure is also introduced on the two sets of segments by using buckets. Each image is divided into a number of non overlapping square windows. To each window is attached the list of segments intersecting it, and to each segment is attached the list of windows it intersects. These buckets give fast access to the segments which are close to a given segment and are used to speed up the processes of generating and propagating the hypotheses.

2.4.2 Defining matches

A match is a pair (L, R) or (R, L) of left and right segments satisfying the epipolar constraint and the feature similarity constraint. The epipolar constraint for line segments implies that homologous segments have at least one analogous point. To make things simpler, this point is chosen to be the midpoint of the segment, i.e we impose that the epipolar line associated with the midpoint of segment L intersects segment R for the match (L, R) to be possibly acceptable. Notice that this definition is not symmetric with respect to L and R which has the advantage of potentially permitting to globally match contours which have been approximated by a different number of segments (see Figure 18).

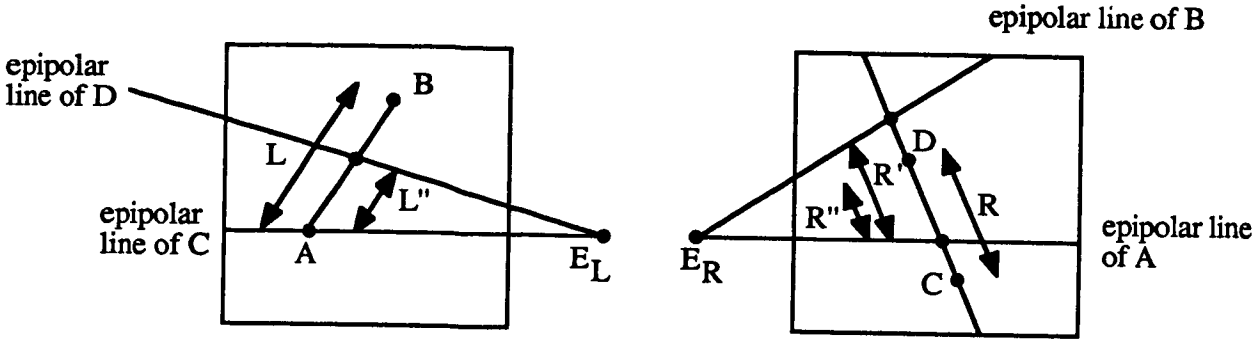


Figure 19: Disparity of two segments

The disparity of a match (L, R) or (R, L) is defined in a symmetric way (see Figure 19). We consider the endpoints A and B of L (C and D of R in the case of a (R, L) match), their corresponding epipolar lines intersect the line supporting R (L) along a segment R' (L'). The intersection R'' of R' and R (L'' for L' and L) is not empty, by definition of an acceptable match; using a similar procedure, the endpoints of R'' yield L'' included in L (R'' included in R). The segments L'' and R'' have all of their points in correspondence. The disparity of the two matches (L, R) and (R, L) is then defined as the average disparity along L''.

2.4.3 Making hypothesis

A number of hypothetical matches are made uniformly over the whole image by randomly selecting segments in the left image. In order to reduce the number of false matches, only segments which are neither too small (poor estimation of the orientation) or too long (likely to be broken in the right image) and with an orientation not too close to that of the epipolar lines are selected. The thresholds for comparing segments features are set quite low at this stage, to eliminate as many false matches as possible.

2.4.4 Verifying hypothesis

Once hypothesis have been made, the algorithm attempts to propagate them by traversing a graph called the disparity graph whose use is to enforce the disparity gradient constraint (to be defined below). Vertexes of this graph are matches (L, R). Two vertexes are connected by an arc if and only if the corresponding disparity gradient (the difference between the disparities of the two matches) is less than a threshold. An important point is that this threshold is variable and depends on the position in 3D space of the line segments reconstructed from the matches. This threshold is set to tolerate a constant variation in depth, therefore it can be considered as a constraint on the roughness of the reconstructed objects. It is computed as follows. Given a match (L, R) and the corresponding disparity $DISP$, let M be the reconstructed point from I_L and I_R (see Figure 20). A maximum variation of depth of ε around M corresponds to points I_{R1} and I_{R2} on the epipolar line of I_L, with disparities $DISP1$ and $DISP2$. A match (L', R') is connected to (L, R) if and only if its disparity $DISP'$ is such that $DISP' - DISP$ lies in the interval $[Min(DISP1 - DISP, DISP2 - DISP),\ Max(DISP1 - DISP, DISP2 - DISP)]$. The propagation of hypothesis can be considered as the building of connected components of the disparity graph. This is implemented by the following two procedures:

```
procedure propagate_left(left_segment, predicted_disparity)
begin
    if not already_visited(left_segment) then
        begin
            (right_segment, actual_disparity):=match_left(left_segment,
            predicted_disparity)
            if not (right_segment=NIL) then
                for each neighbor of right_segment
                    propagate_right(right_segment, actual_disparity)
        end
end

procedure propagate_right(right_segment, predicted_disparity)
begin
    if not already_visited(right_segment) then
        begin
```

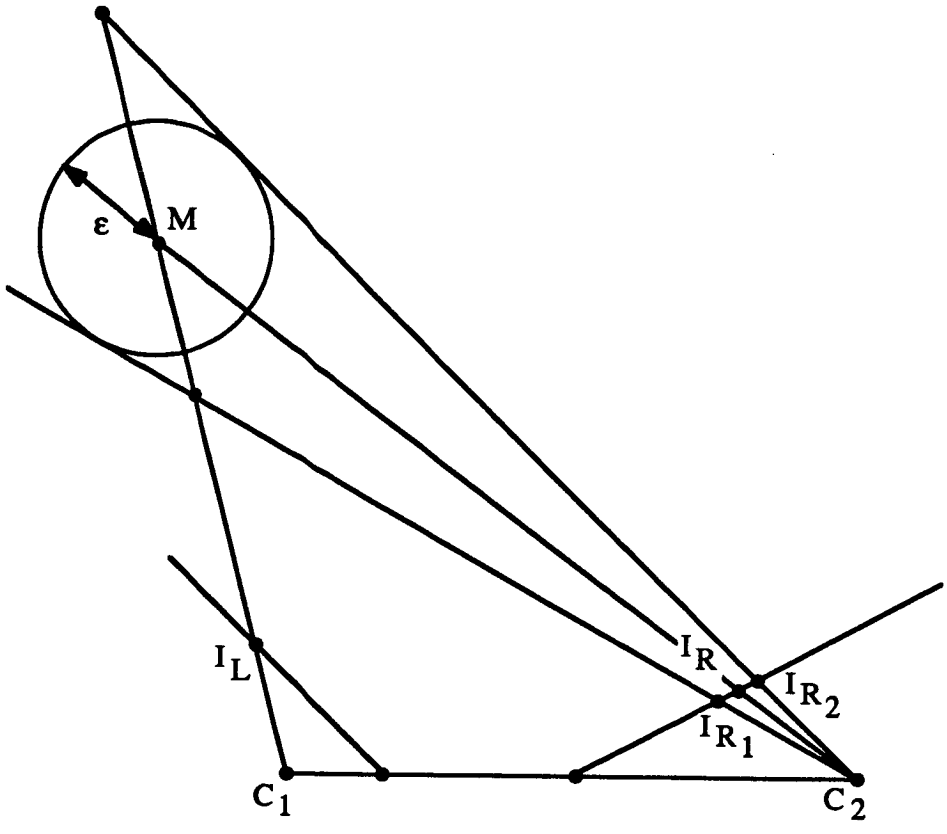


Figure 20: Disparity gradient

(left_segment, actual_disparity):=**match_right**(right_segment, predicted_disparity)
if not (left_segment=NIL) **then**
for each neighbor of left_segment
propagate_left(left_segment, actual_disparity)
end
end

To propagate an hypothesis (L, R) with disparity $DISP$, the procedure **propagate_right** is activated with the parameters R and $DISP$. It first checks whether R has already been visited; if not, it calls the procedure **match_right** which selects among the left segments those whose disparity with R is in the range $DISP1$, $DISP2$, and whose features are sufficiently close to those of R. If several candidates segments remain, the one with disparity closest to $DISP$ is chosen.

If no match is found, the procedure **match_right** returns NIL then **propagate_right**

stops, otherwise when a match L' with disparity $DISP'$ has been found, **propagate_right** calls the symmetric procedure **propagate_left** for each neighbor of L' with the disparity $DISP'$, etc ....

2.4.5 Handling conflicts

A conflict is defined as a pair of matches (L, R) and (L, R') or (L, R) and (L', R). Conflicts occurring in the propagation phase are handled immediately by choosing the best match in terms of the features. Conflicts occuring between two distinct components of the disparity graph are solved by comparing the sizes of the the two components and erasing from the smallest one the conflicting matches. To avoid a dependence on the order in which the components are compared, the size comparison is made on the initial size of the components.

2.5 Adding the planarity constraint

In many human made environments, we are bound to encounter many flat surfaces such as walls, floors, ceilings, doors. Those surfaces are planar. This in turn implies a very strong constraint on the kind of correspondence which occurs between the two images of a stereo pair. In fact it turns out that if the visual features which are matched are emanating from a plane, there is no ambiguity in the matching since there exists an analytic transformation from the left image coordinates to the right image coordinates. This analytic transformation is a function of the rotation and translation from camera 1 camera 2 and of the plane equation.

2.5.1 Analytic Transformation between Left and Right Images

In order to derive this transformation, we use the notations of Section 2 and Figure 21. Let $\mathbf{M}_1$ and $\mathbf{M}_2$ be the perspective transformations for the two cameras, $\mathbf{R}$ the rotation matrix from the left camera to the right, and $\mathbf{t}$ the translation vector. It can be easily shown, that the relationship between $\mathbf{M}_1$ and $\mathbf{M}_2$ is:

$$\mathbf{M}_2 = \mathbf{M}_1 \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

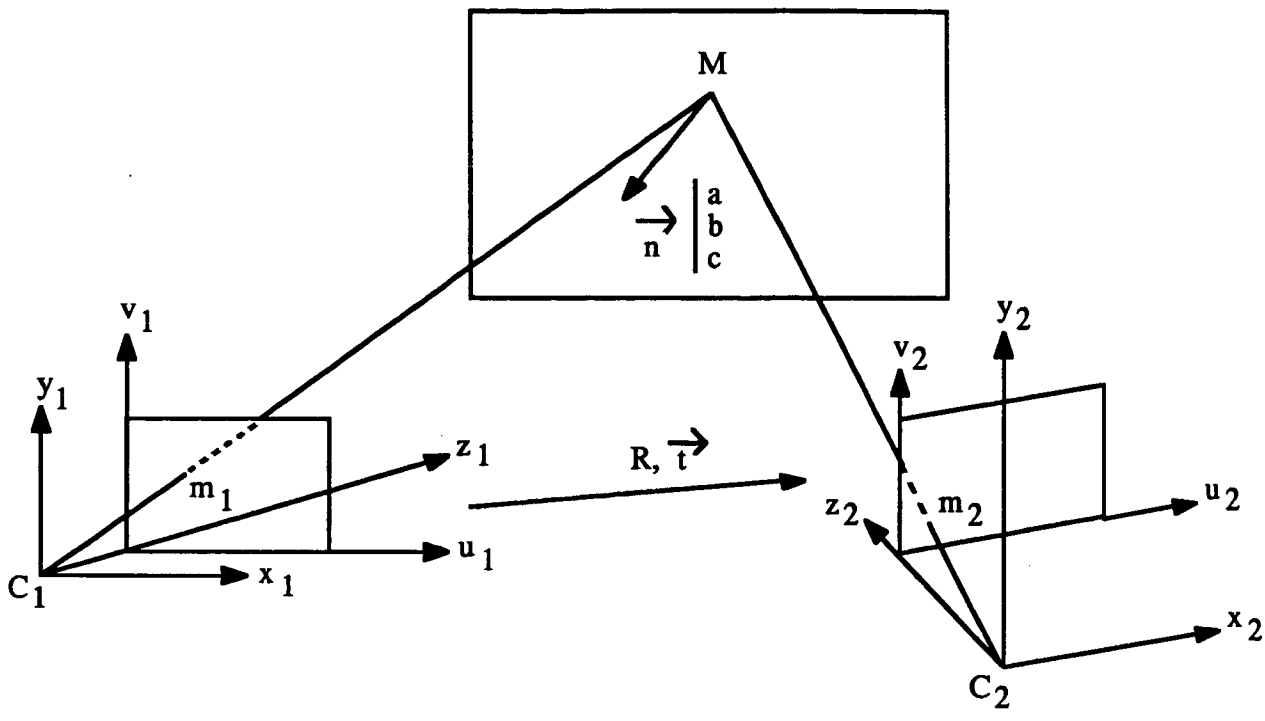


Figure 21: Looking at a plane

Let us suppose now that these two cameras are looking at a plane given by its normal $\mathbf{n} = [a, b, c]^T$ and its distance to the origin d (we assume that the normal is of length 1), and let m_1 of coordinates (u_1, v_1) be a point in the camera plane of camera 1. The line $C_1 m_1$ is described by the points of coordinates $r\mathbf{N}_1$ for $-\infty < r < +\infty$, with $\mathbf{N}_1$ the vector $[u_1, v_1, f]^T$, where f is the focal length.

The value of r corresponding to the intersection of line $C_1 m_1$ with the plane is obtained by writing:

$$\mathbf{n}.r\mathbf{N}_1 = d$$

This yields the object point M. The coordinates of the corresponding point m_2 in the second camera plane are given by:

$$\begin{bmatrix} U_2 \\ V_2 \\ T_2 \end{bmatrix} = \mathbf{M}_2 \begin{bmatrix} (d/\mathbf{n}.\mathbf{N}_1)\mathbf{N}_1 \\ 1 \end{bmatrix}$$

This is the same as:

$$\begin{bmatrix} U_2 \\ V_2 \\ T_2 \end{bmatrix} = \mathbf{M}_2 \begin{bmatrix} d\mathbf{N}_1 \\ \mathbf{n}.\mathbf{N}_1 \end{bmatrix}$$

Since $\mathbf{n}.\mathbf{N}_1$ is equal to $au_1 + bv_1 + cf$, this can be rewritten as:

$$\begin{bmatrix} U_2 \\ V_2 \\ T_2 \end{bmatrix} = \mathbf{M}_2 \begin{bmatrix} d\mathbf{I} & \\ a & b \quad c \end{bmatrix} \begin{bmatrix} U_1 \\ V_1 \\ T_1 \end{bmatrix} \tag{2}$$

where $\mathbf{I}$ is the 3×3 identity matrix. Therefore, the relationship between the point m_1 and the point m_2 is linear in projective space when the object point M moves in a plane. Let us rewrite it more compactly as:

$$\mathbf{m}_2 = \mathbf{A}\mathbf{m}_1 \tag{3}$$

Equations 2 and 3 describe completely the relationship between features in the left and right images.

Two questions can be asked at this point. First, once we know matrix $\mathbf{A}$, can we recover the plane equation, and second how can we use this information to help solve the stereo problem.

2.5.2 Recovering the plane equation

Let us answer the first question first and denote by β_i, $i = 1, \ldots, 4$, the 4 column vectors of size 3 of matrix $\mathbf{M}_2$. We have to solve the following equation:

$$\mathbf{M}_2 \begin{bmatrix} d\mathbf{I} & \\ a & b \quad c \end{bmatrix} = \mathbf{A}$$

where the unknowns are the plane parameters a, b, c, d. This is a linear system in those unknowns which can be solved by classical mean-square techniques. Here we find, denoting by $\mathbf{p}$ the vector $[a, b, c, d]^T$ and by $\mathbf{a}_i$, $i = 1, \ldots, 3$, the three column vectors of size 3 of matrix $\mathbf{A}$:

$$\mathbf{M}\mathbf{p} = \mathbf{a}$$

and therefore:

$$\mathbf{p} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{a}$$

where $\mathbf{M}$ is the 9×4 matrix:

$$\begin{bmatrix} \beta_4 & 0 & 0 & \beta_1 \\ 0 & \beta_4 & 0 & \beta_2 \\ 0 & 0 & \beta_4 & \beta_3 \end{bmatrix}$$

and $\mathbf{a}$ is the 9-dimensional vector $[\mathbf{a}_1^T, \mathbf{a}_2^T, \mathbf{a}_3^T]^T$. We do not need to take into account the constraint that the normal to the plane is of unit norm since matrices $\mathbf{M}$ and $\mathbf{A}$ are defined up to a scale factor.

2.5.3 Estimating matrix A

Another important related question is the estimation of matrix $\mathbf{A} = [a_{ij}]$, i, $j = 1, 2, 3$ from matches between the left and right images. So far we have assumed that we were only using points, but as we show next we can also use line segments or rather the infinite lines supporting those line segments. Let us consider the two cases separately.

Matching points: from equation 3 we get:

$$T_2 = T_1(a_{31}u_1 + a_{32}v_1 + a_{33}) \tag{4}$$

and therefore:

$$a_{11}u_1 + a_{12}v_1 + a_{13} = u_2(a_{31}u_1 + a_{32}v_1 + a_{33}) \tag{5}$$

$$a_{21}u_1 + a_{22}v_1 + a_{23} = v_2(a_{31}u_1 + a_{32}v_1 + a_{33}) \tag{6}$$

Since $\mathbf{A}$ is defined up to a scale factor, we can impose a condition on the coefficients a_{ij}. A possible and often used condition is to set $a_{33} = 1$. It must be kept in mind that this prevents of course a_{33} to be equal to zero. This has a corresponding geometric interpretation: if $a_{33} = 0$, the condition for the point m_2 to be at infinity can be written, according to equation 4, $a_{31}u_1 + a_{32}v_1 = 0$. This is the equation of a line in the first camera plane that goes through the origin. This line is the image by the first camera of the line intersection of the plane being looked at and the focal plane of the second camera. Therefore, if we set $a_{33} = 1$, this prevents the previous situation to happen, i.e the image by the first camera of this line cannot go through the origin. In practice, this is not a problem. Let us now define:

$$\begin{aligned} \mathbf{b} &= [u_1, v_1, 1, 0, 0, 0, -u_2u_1, -u_2v_1]^T \\ \mathbf{c} &= [0, 0, 0, u_1, v_1, 1, -v_2u_1, -v_2v_1]^T \\ \mathbf{a} &= [a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}]^T \end{aligned}$$

Equations 5 and 6 can then be rewritten as:

$$\mathbf{b}^T\mathbf{a} = u_2$$

$$\mathbf{c}^T\mathbf{a} = v_2$$

This is for one pair of points, for n pairs the corresponding meansquare problem is to minimize:

$$C = \sum_n (\mathbf{b}_n^T\mathbf{a} - u_{2n})^2 + (\mathbf{c}_n^T\mathbf{a} - v_{2n})^2$$

or:

$$C = \mathbf{a}^T\mathbf{H}\mathbf{a} - 2\mathbf{h}^T\mathbf{a} + r$$

where:

$$\begin{aligned} \mathbf{H} &= \sum(\mathbf{b}_n\mathbf{b}_n^T + \mathbf{c}_n\mathbf{c}_n^T) \\ \mathbf{h} &= \sum(u_{2n}\mathbf{b}_n + v_{2n}\mathbf{c}_n) \\ r &= \sum(u_{2n}^2 + v_{2n}^2) \end{aligned}$$

The solution $\mathbf{a} = 2\mathbf{H}^{-1}\mathbf{h}$ can be computed iteratively using standard recursive least squares techniques.

Matching lines: it turns out that this is very similar to the matching of points, something to be expected since in planar projective geometry points and lines play the same role. Indeed, let L_1 be a straight line in the first camera plane defined by its equation:

$$\mathbf{n}_1^T\mathbf{m}_1 = 0$$

and let L_2 be the corresponding line in the second camera plane defined by its equation:

$$\mathbf{n}_2^T\mathbf{m}_2 = 0$$

Using equation 3 we get:

$$\mathbf{n}_1 = \mathbf{A}^T\mathbf{n}_2 \tag{7}$$

Therefore the previous procedure can also be applied to lines modulo the difference between equations 3 and 7.

How many matches are necessary in order to estimate matrix $\mathbf{A}$? since it depends only on eight coefficients, 4 points in general position, i.e such that not three of them are

aligned are necessary, or two segments. Two segments can be seen to yield four points in general position by choosing two on each line supporting the segments, and using the epipolar geometry to compute their correspondents in the second image.

It is now possible to modify the algorithm of Section 2.4 in such a way that it takes into account the planarity constraint. The result is an algorithm that solves the stereo matching problem and at the same time finds planes.

An hypothesis is defined to be two pairs (L_1, R_1) and (L_2, R_2) of acceptable matches in the sense of the previous Section which are also such that the corresponding 3D lines are coplanar. If this hypothesis is correct, then it can be propagated by first estimating the corresponding matrix $\mathbf{A}$, and second applying it to the unmatched neighbors of L_1 and L_2. Let L be such a neighbor, then $\mathbf{A}L$ is a segment in the right image which should find a matching segment with similar features in its immediate neighborhood if the hypothesis is right and none if it is not.

When such a match has been found, estimation of matrix $\mathbf{A}$ can be updated and then applied to more segments until no more matches can be found. Once a plane has been found by propagating an hypothesis, others can be searched for by considering different hypothesis involving yet unmatched segments. Results are shown in Figures 22 and 23.

2.6 Using three cameras

The idea of using three cameras ([GPSL86,GDS86,II86a,II86b,OWI86,PH86,YAC86], and [YKK86,AL87] is that of making fuller use of the epipolar constraint. Indeed, let us consider the three cameras system of Figure 24. We give the idea for a point token, understanding that it can be extended to other tokens as well.

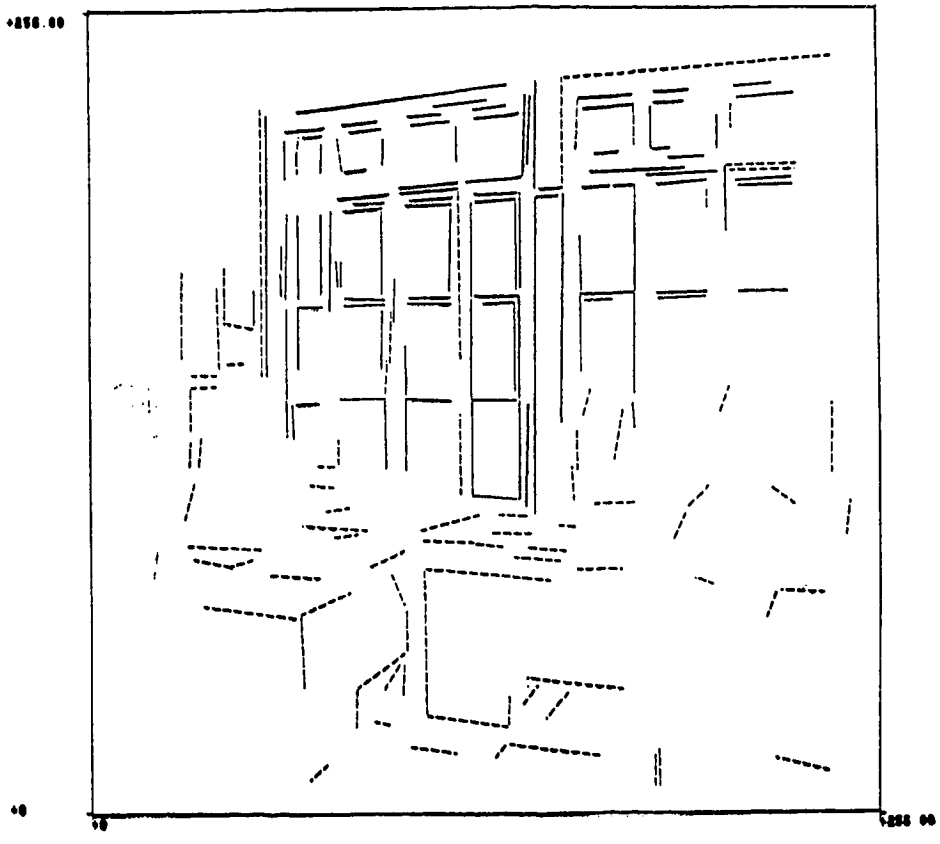
Let $\mathbf{m}_1$ be a point in camera 1. Its potential matches in cameras 2 and 3 are located on the epipolar lines E_{12} and E_{13} of $\mathbf{m}_1$. Let $\mathbf{m}_2$ be a candidate match on E_{12}; $\mathbf{m}_2$ has an epipolar line E_{23} in camera 3 which intersects E_{13} at a point $\mathbf{m}_3$. Therefore, if the match $(\mathbf{m}_1, \mathbf{m}_2)$ is correct, there should be an image feature at $\mathbf{m}_3$ which is similar to that at $\mathbf{m}_1$ and $\mathbf{m}_2$. This test is quite simple when the epipolar geometry is known and drastically reduces the complexity of the search for correspondences.

The algorithm is then as follows (for line segments):

For each segment S_1 in image 1

Find S_2 in image 2

a)



b)

Figure 22: a) Stereo pair of an office scene. b) Segments found to be in the window plane.

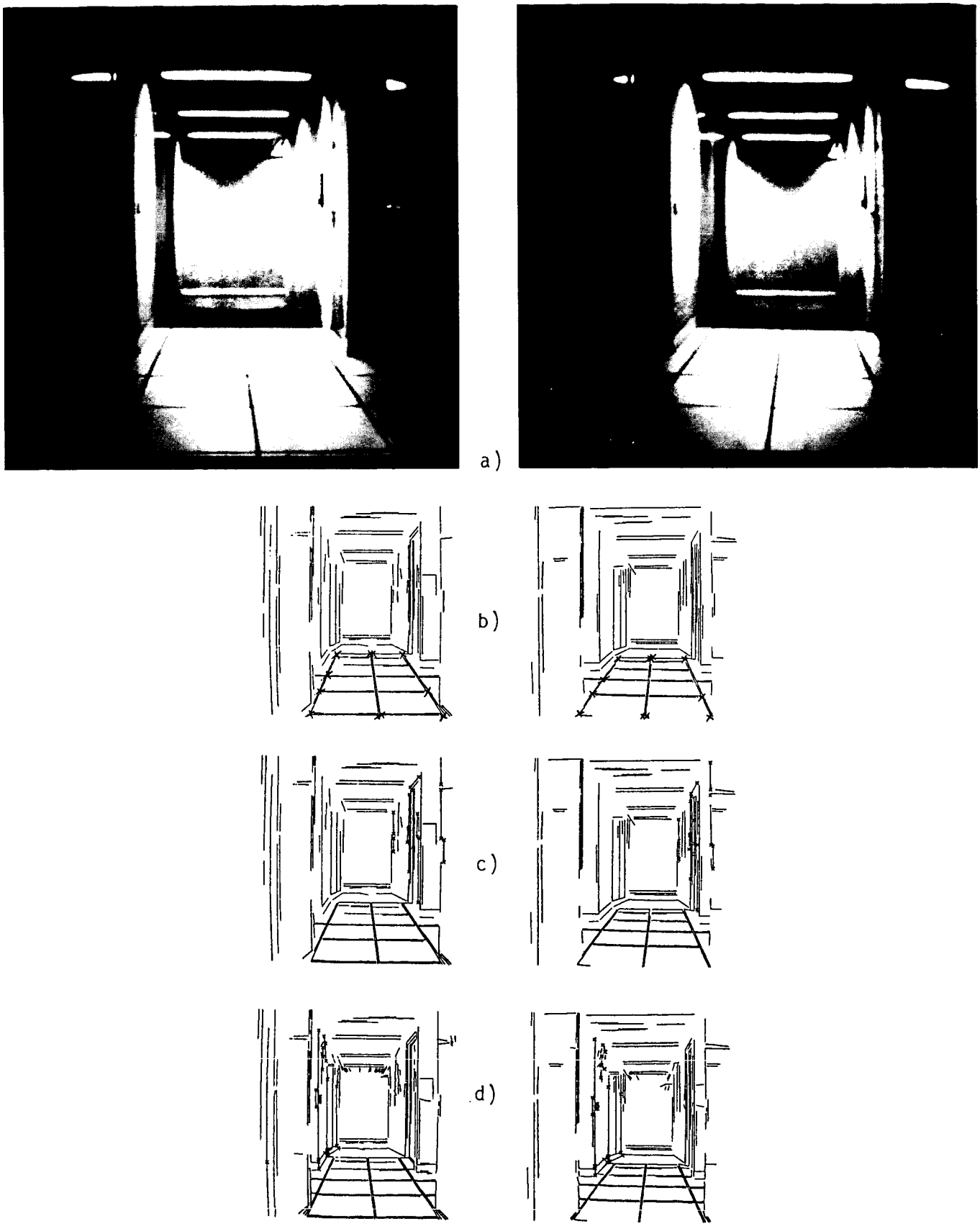


Figure 23: a) Stereo pair of a hallway. b) Points found to be on the floor plane. c) Points found to be on the left wall plane. d) Points found to be on the right wall plane.

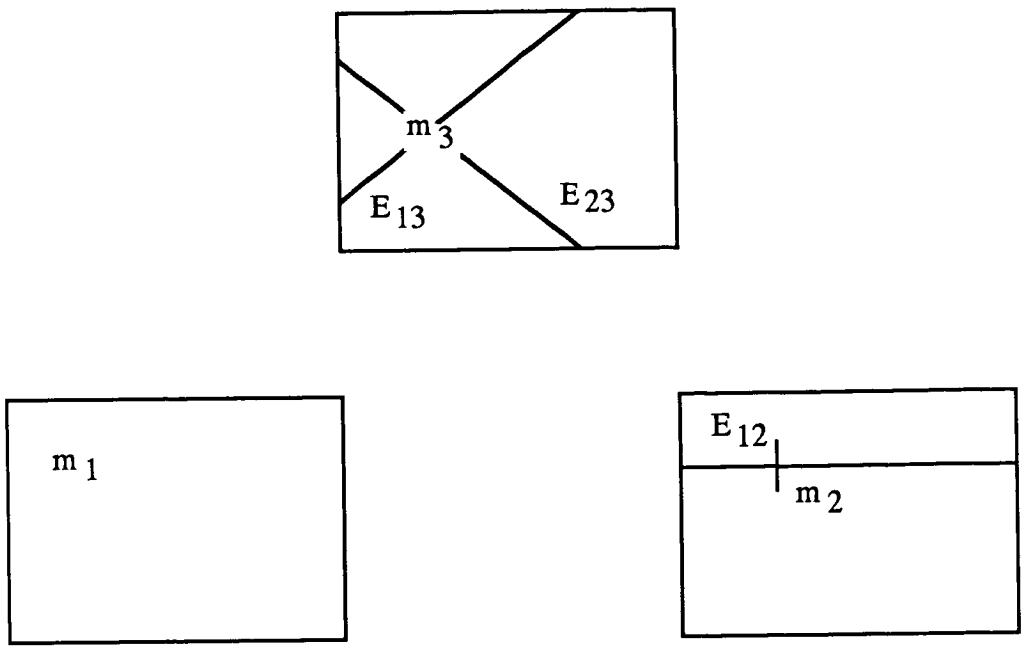


Figure 24: Three camera stereo

Intersecting E_{12}
Similar to S_1
For each such S_2 in image 2
Find S_3 in image 3
Intersecting the intersection of E_{13} and E_{23}
Similar to S_1 and S_2

The verification phase of Section 2.4.4 has now disappeared and is replaced by a simple elimination of potential false triplets (S_1, S_2, S_3) by applying to the reconstructed 3D segments a procedure enforcing local compatibility [AL87]. The result is a more accurate, more reliable, and faster algorithm. Figures 25, 26, and 27 show some results.

2.7 Motion and Structure from Motion

Motion analysis is also a way of recovering 3D data which we believe is important and may not have been exploited enough in the past. The motion work we present here is a simplification of the general problem in that we assume that only the camera is moving and that the environment is otherwise static. Under this simplifying assumption, it is well known that the motion of the camera and the 3D structure of the environment can be

Figure 25: Stereo triplet of images

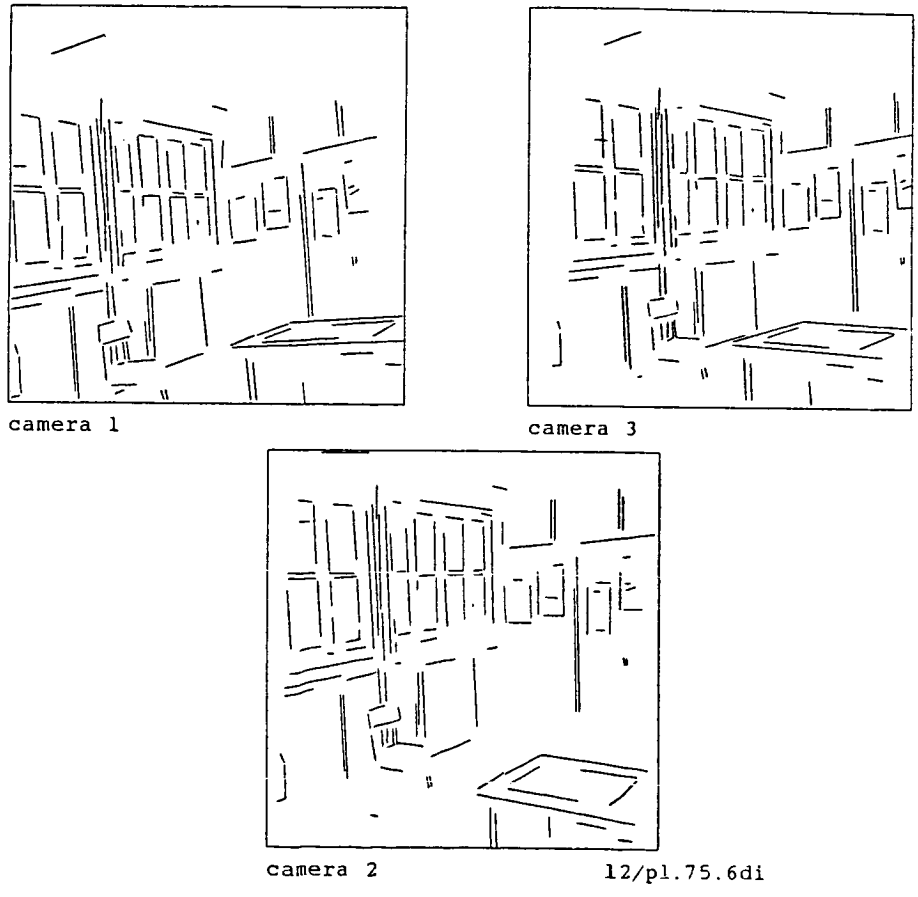


Figure 26: Matched segments in the three images

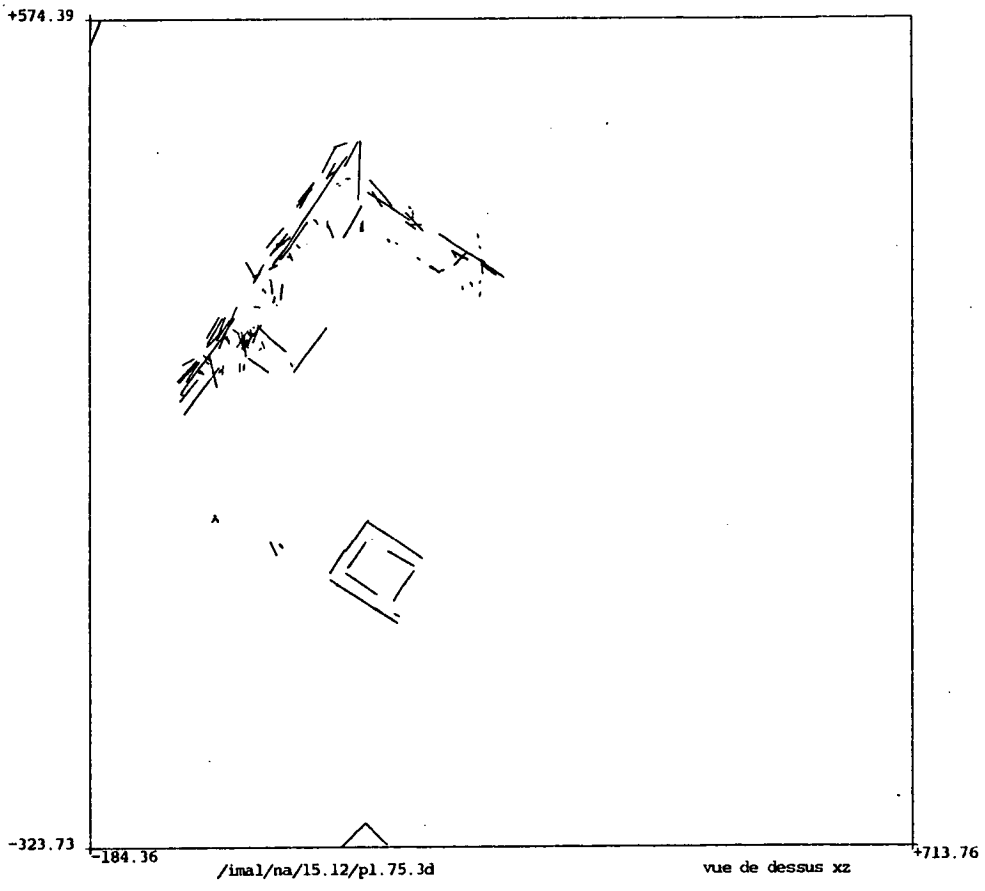


Figure 27: Orthographic projection in a horizontal plane of the reconstructed 3D segments

recovered up to a scale factor. This scale factor can then be recovered if the absolute size of some object in the scene is known.

Traditionnally there have been two main approaches to the analysis of motion. The first approach consists in computing the optical flow or image displacement field, assume that it is the projection of the actual 3D motion field and recover the 3D field from it. The second approach consists in matching tokens between frames and recovering 3D motion and structure from those matches. For an excellent recent review, see Nagel [NAG86].

Problems with the first approach are:

1. The flow cannot be recovered entirely at every pixel (window effect, points with no gradient) [HIL84].

2. The optical flow is usually not the projection of the 3D field [VP87].

The main problem with the second approach is that finding the matches may not be simple. A general problem with both approaches is that the results are believed to be quite sensitive to noise and errors. Some recent work conducted at INRIA [FLT87] seems to indicate that 3D motion and structure can be robustly estimated from token matches. This work points to the fact that the sensitivity to noise is not as big a problem as we used to think it was. We also think that in most Robotics applications, given fast enough processing, the matching problem becomes essentially trivial because objects have not moved much from one frame to the next if the sampling frequency along the time axis is high enough.

We have investigated two kinds of tokens: points and lines. Points are corners, i.e intersections of lines. Matching two points yields the whole 2D displacement (optical flow) whereas matching two lines does not.

2.7.1 Point matches

The problem of recovering motion and structure from point matches has been studied by Longuet-Higgins [LON84,LON81] who proposed an algorithm based on 8 point matches. This algorithm fails when the points are not in general configuration and in particular when they are coplanar. It is also quite sensitive to noise. In the planar case, a method developed by Tsai and Huang [TH82] requires 4 point matches. It has also been reported to be quite sensitive to noise.

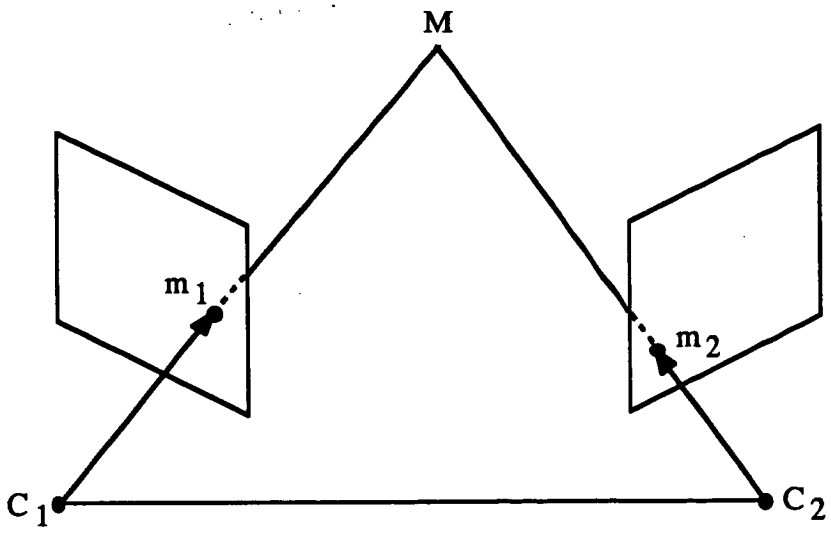


Figure 28: Planarity constraint in motion estimation

In the two cases, the use of mean-square techniques (linear and nonlinear) has allowed us to become quite insensitive to noise. We discuss here the case of nonplanar points. The interested reader is referred to [FL87,FLT87] for the case of planar points.

The problem with points in general positions has been solved by Longuet-Higgins [LON84] in the exact case, assuming no noise. When noise is present, the method is quite sensitive [HUA86]. We propose here a method which can cope more robustly with the noise problem. It is based on the exploitation of the planarity constraint used by Longuet-Higgins.

Let us look at Figure 28 where m_1 and m_2 are the images of the physical point M in the two cameras. The three vectors $\mathbf{C_1m_1}$, $\mathbf{t}$, and $\mathbf{C_2m_2}$ are coplanar and therefore their determinant is zero. What we can measure are the coordinates of $\mathbf{C_1m_1}$ and $\mathbf{C_2m_2}$ in the coordinate systems $C_1x_1y_1z_1$ and $C_2x_2y_2z_2$ of Figure 21. $\mathbf{C_2m_2}$ expressed in the coordinate system 1 is equal to $\mathbf{RC_2m_2}$. Therefore, we have, in the coordinate system 1:

$$det(\mathbf{C_1m_1},\ \mathbf{t},\ \mathbf{RC_2m_2}) = 0$$

Or, more geometrically:

$$\mathbf{C_1m_1}.(\mathbf{t} \wedge \mathbf{RC_2m_2}) = 0 \tag{8}$$

Introducing the antisymmetric matrix $\mathbf{T}$:

$$\mathbf{T} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

equation 8 can be rewritten as:

$$\mathbf{C_1 m_1}^T \mathbf{E}\, \mathbf{C_2 m_2} = 0 \tag{9}$$

with:

$$\mathbf{E} = \mathbf{TR}$$

Since only the direction of t can be recovered, $\mathbf{E}$ is defined up to a scale factor. We normalize things by assuming that $\|\mathbf{t}\| = 1$. This implies that $\|\mathbf{E}\|^2 = 2$.

Notice that equation 9 is a linear equation in the coefficients of matrix $\mathbf{E}$. Denoting its row vectors by $\mathbf{X_1}^T$, $\mathbf{X_2}^T$, $\mathbf{X_3}^T$ and by $\mathbf{X}$ the 9×1 vector $[\mathbf{X_1}^T, \mathbf{X_2}^T, \mathbf{X_3}^T]^T$, we can write equation 9 as:

$$\mathbf{a}^T\mathbf{X} = 0$$

where:

$$\mathbf{a^T} = [x_1x_2, x_1y_2, x_1, y_1x_2, y_1y_2, y_1, x_2, y_2, 1]^T$$

and the x_i's and y_i's are the coordinates of $\mathbf{C_1 m_1}$ and $\mathbf{C_2 m_2}$ in coordinate systems 1 and 2 (we assume a focal length of 1).

If we have n matches, we have n such equations, and a linear system:

$$\mathbf{A_n X} = 0 \tag{10}$$

where $\mathbf{A_n}$ is a $n \times 9$ matrix. Longuet-Higgins solves this system for $n = 8$ when the rank of $\mathbf{A_8}$ is 8. The null space is then of dimension 1 and $\mathbf{X}$ is the vector of norm $\sqrt{2}$ in the null space. Degenerate cases occur when $rank(\mathbf{A_8}) < 8$ and are analyzed in Longuet-Higgins [LON81] and Zhuang and Haralick [ZH85]. We discuss in [FLT87] the case where the 3D points are coplanar which also yields a degenerate matrix $\mathbf{A_8}$.

When $rank(\mathbf{A_n}) = 8$, our technique works in three steps:

1. Compute $\mathbf{X}$: solve equation 10 by meansquare.

   Noticing that $\|\mathbf{X}\|^2 = 2\|\mathbf{t}\|^2 = 2$, this is equivalent to solving:

   $$\min_{\mathbf{X}} \|\mathbf{A_n X}\|^2 \quad \text{subject to} \quad \|\mathbf{X}\|^2 = 2$$

   which is a standard problem.

2. Compute t: we then notice [NAG86] that $\mathbf{t}^T\mathbf{E} = \mathbf{t}^T\mathbf{TR} = \mathbf{0}$. Therefore we obtain t as the solution of:

$$\min_{\mathbf{t}} \|\mathbf{E}^T\mathbf{t}\|^2 \quad \text{subject to} \quad \|\mathbf{t}\|^2 = 1$$

another standard problem

3. Compute **R**: **R** is then computed as the solution of:

$$\min_{\mathbf{R}} \|\mathbf{E} - \mathbf{TR}\|^2 \quad \text{subject to} \quad \mathbf{R}^T\mathbf{R} = \mathbf{I}$$

this is easily solved using for example the quaternion representation of rotation matrixes (see Section 4, Appendix B, and [FH86,FT86]).

We have tested this algorithm on synthetic and real data. The image of the 3D test scene we have used is shown in Figure 29. It is made of 10 points situated at a distance of 4 meters from our camera. The size of the object is also about four meters. The perspective transformation matrix we use to project the 3D points has been obtained by calibration of a real camera. We have simulated a number of camera displacements and noise in pixel measurements. The resolution is 512×512 pixels, and noise varies from 1 to 5 pixels. Once we have estimated the rotation **R** and the direction of the translation t, we can reconstruct the 3D points by meansquare and project them back in the two retinas before and after motion. We then compute the average distance between the actual and the reconstructed pixels. The sum of these two measurements constitute our measure of quality.

Table 1 shows some typical results for two techniques (the original Longuet-Higgins technique, improved to take into account the noise when estimating **E**, and ours). Two main remarks can be made at this point:

1. both techniques appear to be quite robust to quite large noise distortions contrarily to what has been reported in the literature so far.

2. our method performs at least as well as the Longuet-Higgins one, and significantly better in many cases.

Figure 30 shows the projected reconstructed points in the two retinas for our method in the case of line 2 of Table 1.

We have also experimented with real data which have been used in camera calibration experiments. We have taken four stereo views of a special pattern displayed in Figure 31

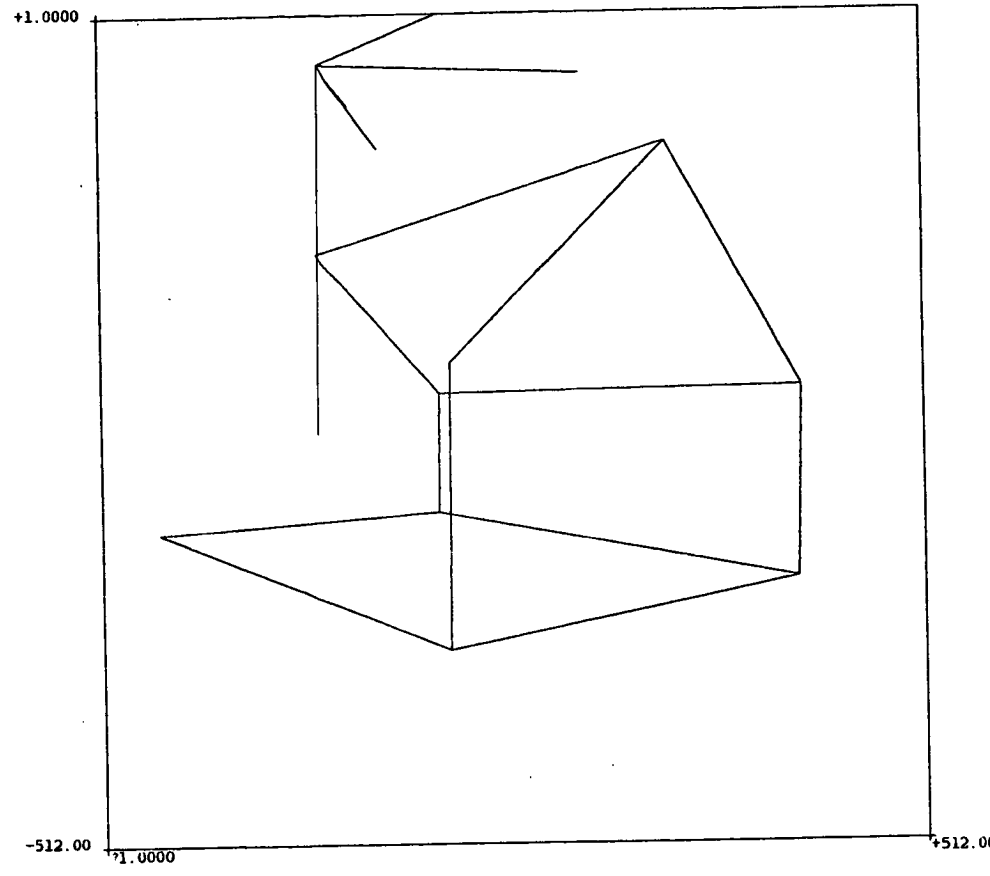


Figure 29: 3D test scene

ϑ	t	∂p	av	sd	ϑ'	$\vartheta\%$	α	β	
					5.89	17.8	24.59	26.24	(1)
5	10	1	57.8	7.2	5.36	7.2	11.48	27.58	(2)
					5.21	4.2	2.60	4.12	(3)
					15.18	203.6	42.21	4.53	(1)
5	50	1	91.8	14.3	5.72	14.5	6.78	1.78	(2)
					5.25	5.0	3.31	0.78	(3)
					11.03	120.7	37.31	3.09	(1)
5	100	1	139.6	31.1	5.74	14.8	6.30	0.82	(2)
					5.25	5.1	3.38	0.38	(3)
					6.06	21.2	47.34	72.43	(1)
5	10	5	57.7	8.1	5.04	0.8	16.94	72.61	(2)
					5.14	2.8	8.61	71.91	(3)
					20.01	300.2	115.43	24.59	(1)
5	50	5	91.5	14.7	8.37	67.3	37.77	11.48	(2)
					5.99	19.8	10.12	4.31	(3)
					55.90	1018	121	23.2	(1)
5	100	5	139.4	31.1	9.09	82	22.3	5.06	(2)
					6.17	23.4	11.81	2.11	(3)
					34.6	130.7	35.6	6.68	(1)
15	50	5	189.1	20.4	19.5	30.3	8.5	7.84	(2)
					15.05	0.14	2.78	5.05	(3)
					25.15	67.6	24.98	4.19	(1)
15	100	5	232.3	30.3	19.73	31.6	5.37	1.93	(2)
					15.24	1.6	2.63	2.64	(3)

axis of rotation: (1,1,1)

direction of translation: (1,0,0)

ϑ = angle of rotation (deg)

t = magnitude of exact translation (cm)

∂p = measurement noise (pixels)

av = average of image displacement (pixels)

sd = standard deviation of image displacement (pixels)

ϑ' = angle of computed rotation

$\vartheta\%$ = rotation angle error (in %)

α = angle between exact and computed axes of rotation

β = angle between exact and computed direction of translation

(1) = improved Longuet-Higgins

(2) = new linear technique

(3) = reconstruction and reprojection

Table 1: Synthetic data

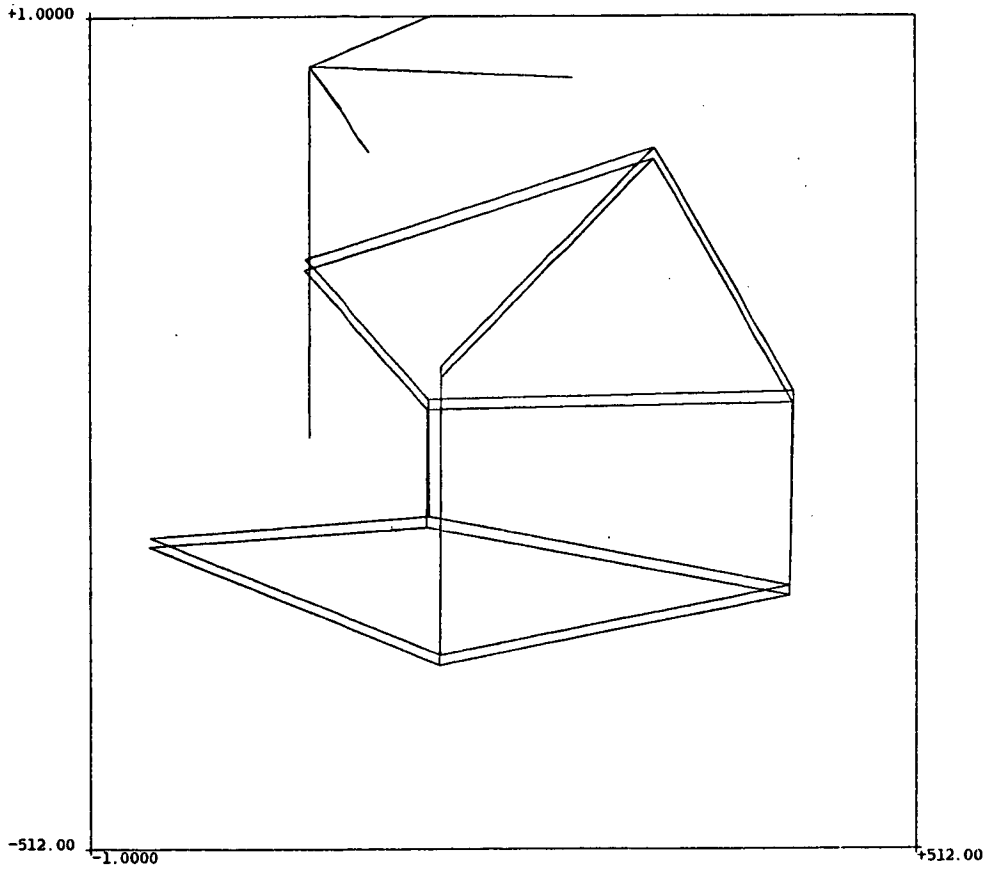


Figure 30: Projected reconstructed points

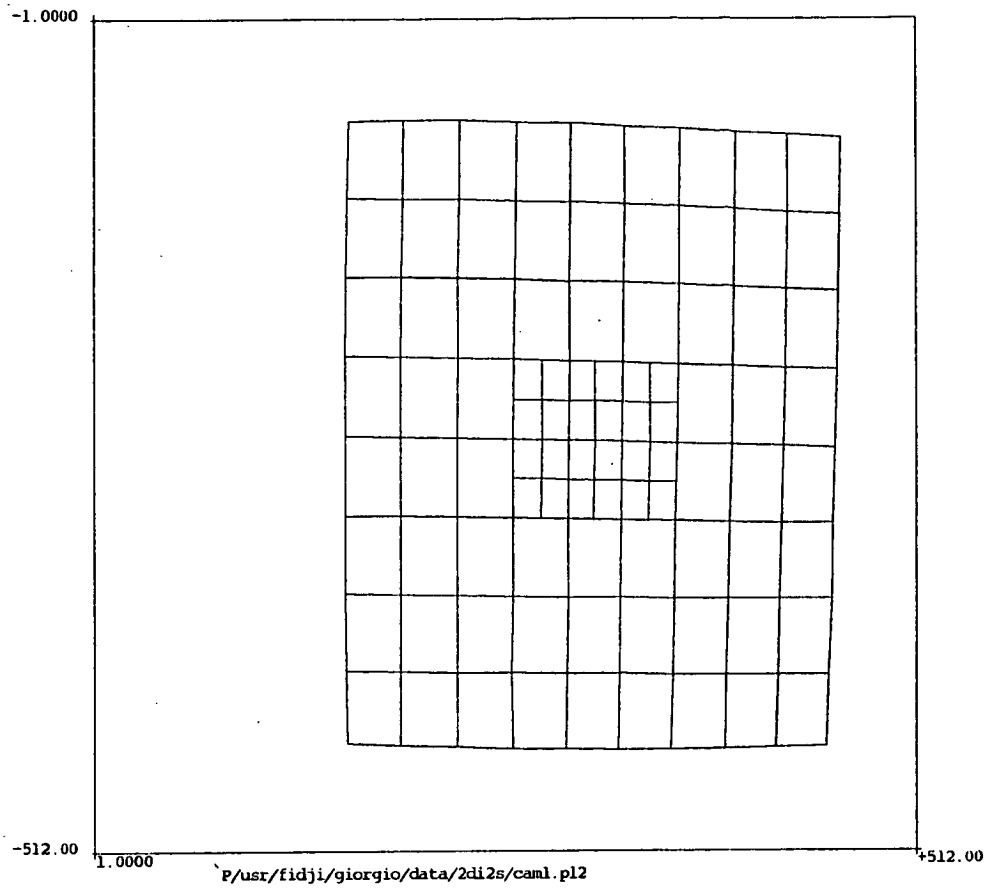


Figure 31: Special grid pattern

made of lines painted black on a sheet of white paper at distances of 3, 3.5, 4, and 4.5 meters. We have extracted automatically those lines from the four images and computed with great accuracy the pixel coordinates of their intersections, thus yielding 113 points in each image. Our method of stereo calibration described in [FT86] gives us the displacement (rotation $\mathbf{R}$ and translation t) from camera 1 to camera 2 by using at least two planes of data. We have then used our motion estimation technique on the points in the other two planes and estimated $\mathbf{R}$ and the direction of translation t. Results are shown in Table 2 for two types of camera. They show that this technique for determining camera motion yields excellent results.

Remarks: The coplanarity constraint of equation 8 is only one way of approaching the problem. There are several other possibilities which yield more complicated equations but may in practice yield better and more robust results. We discuss now two such possibilities.

	θ	t	av	sd	θ'	$\theta\%$	α	β	
					3.36	3.3	0.96	0.93	(1)
I2S	3.48	8	13.4	6.2	3.37	3.1	0.75	0.91	(2)
					3.47	0.2	0.81	0.70	(3)
					14.64	175	58.1	7.25	(1)
I2S	5.32	44	20.6	5.2	5.13	3.6	0.41	0.76	(2)
					5.14	3.5	0.66	0.31	(3)
					4.60	2.0	4.16	2.18	(1)
I2S	4.69	37	24.9	7.2	4.58	2.4	2.40	2.17	(2)
					4.81	2.5	0.53	0.92	(3)
					9.55	1.5	3.10	0.52	(1)
Pul	9.71	53	26.2	16.4	9.68	0.2	0.32	0.02	(2)
					9.69	0.2	0.80	0.008	(3)
					9.43	3.0	1.36	0.22	(1)
Pul	9.15	49	20.7	12.2	9.06	1.0	0.61	0.22	(2)
					9.03	1.3	0.22	0.23	(3)
					8.13	1.6	2.74	0.31	(1)
Pul	8.27	45	18.8	12.4	8.14	1.5	0.65	0.01	(2)
					8.29	0.25	0.02	0.02	(3)

θ = angle of reference rotation (deg)
t = magnitude of reference translation (cm)
av = average of image displacement (pixels)
sd = standard deviation of image displacement (pixels)
θ' = angle of computed rotation
$\theta\%$ = rotation angle error (in %)
α = angle between reference and computed axes of rotation
β = angle between reference and computed direction of translation
(1) = improved Longuet-Higgins
(2) = new linear technique
(3) = reconstruction and reprojection
I2S = I2S CCD camera
Pul = Pulnix CCD camera

Table 2: Real data

The first is to consider the shortest distance between the lines C_1m_1 and C_2m_2 as a function of $\mathbf{R}$ and $\mathbf{t}$. It can be easily proved that

$$d = | \frac{\mathbf{C_1m_1}.(\mathbf{t} \wedge \mathbf{RC_2m_2})}{\|\mathbf{C_1m_1} \wedge \mathbf{RC_2m_2}\|} |$$

therefore minimizing the sum over the pairs of matched points of the d^2 is the same as minimizing $\|\mathbf{A_nX}\|^2$ but with the normalization by $\|\mathbf{C_1m_1} \wedge \mathbf{RC_2m_2}\|$ which is the sine squared of the angle between the lines C_1m_1 and C_2m_2.

The second is to minimize the image reconstruction error as follows. Let us denote by Q_1 and Q_2 the endpoints on C_1m_1 and C_2m_2 of their common perpendicular (see Figure 4), and suppose we choose their midpoint P as the reconstructed 3D point from m_1 and m_2. An alternate candidate is the point obtained by solving for the intersection of C_1m_1 and C_2m_2 by meansquare. We can then project P as m_1' and m_2' on the two retinas and compute the distances $d(m_1, m_1')$ and $d(m_2, m_2')$ as functions of $\mathbf{R}$ and $\mathbf{t}$. Summing those squared distances over all pairs of matched points yields a criterion which can be minimized. This criterion weighs differently points which are far and points which are close since it accepts a larger 3D reconstruction error for far away points than for close points. We have experimented with this idea and the preliminary results are more accurate and robust to noise than the ones obtained by the previous technique [TF87]. Results also appear in Tables 1 and 2. A similar idea has recently been used by Harris [Har87].

2.7.2 Line matches

Lines are tokens which can be extracted very reliably from many kinds of images and are therefore good candidates as a basis for matches. Some work in this area has been reported by Mitiche et. al and Liu and Huang [MSA86a,MSA86b,LH86b,LH86a]. The difference between the two approaches consists in the order in which the problems are solved. Mitiche et. al first solve for structure and then for motion, Liu and Huang first solve for motion and recover structure next. We have extended the work of Liu and Huang and shown that fairly robust estimates of motion and structure could be obtained from line matches [FLT87].

The characteristic of line matches when compared to point matches is that two views are not sufficient but three are needed to recover both motion and structure. Indeed, as shown in Figure 32, matching l and l' simply defines a 3D line L but puts no constraint on C_1 and C_2 (compare with Figure 28). If we now consider Figure 33, the fact that l, l',

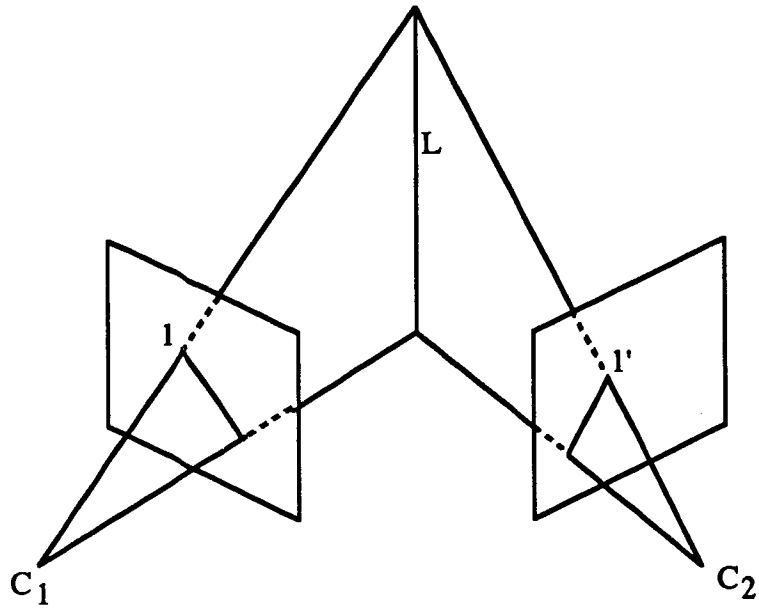


Figure 32: Matching two lines in two frames does not put any constraint on the motion

and l'' are the images at three different times of the same 3D line L imposes constraints on the motion of the camera between positions 1, 2, and 3.

These constraints have their roots in the fact that the three planes defined by l, l', and l'', and the camera optical centers meet along L: they belong to the same pencil of planes. Given a plane of equation $ax + by + cz + d = 0$, we associate a 4-dimensional vector $\mathbf{P} = [a, b, c, d]^T$. Expressing the fact that the three planes belong to he same pencil is equivalent to saying that the 4×3 matrix formed by their associated vectors $\mathbf{P_1}, \mathbf{P_2}$, and $\mathbf{P_3}$ is of rank less than or equal to two. It can be shown that this provides two constraints on the two motions. Since the two motions depend on 11 parameters (6 for the two rotations and 5 for the two translations), a simple counting argument shows that in order to hope to solve for the unknowns, at least 6 line matches are necessary. Because the constraints on the rotations are nonlinear, there is no guarantee that the solution is unique or even that there is a finite number of solutions. We have implemented the motion computation using the Extended Kalman Filtering approach which we briefly mention in Section 4.3 and is developed in [AF87].

We have also implemented this technique both on synthetic and real data. Our synthetic data is the same as the one used in the previous experiment, except that instead of using the points, we have used some of the lines defined by them. Again, we have

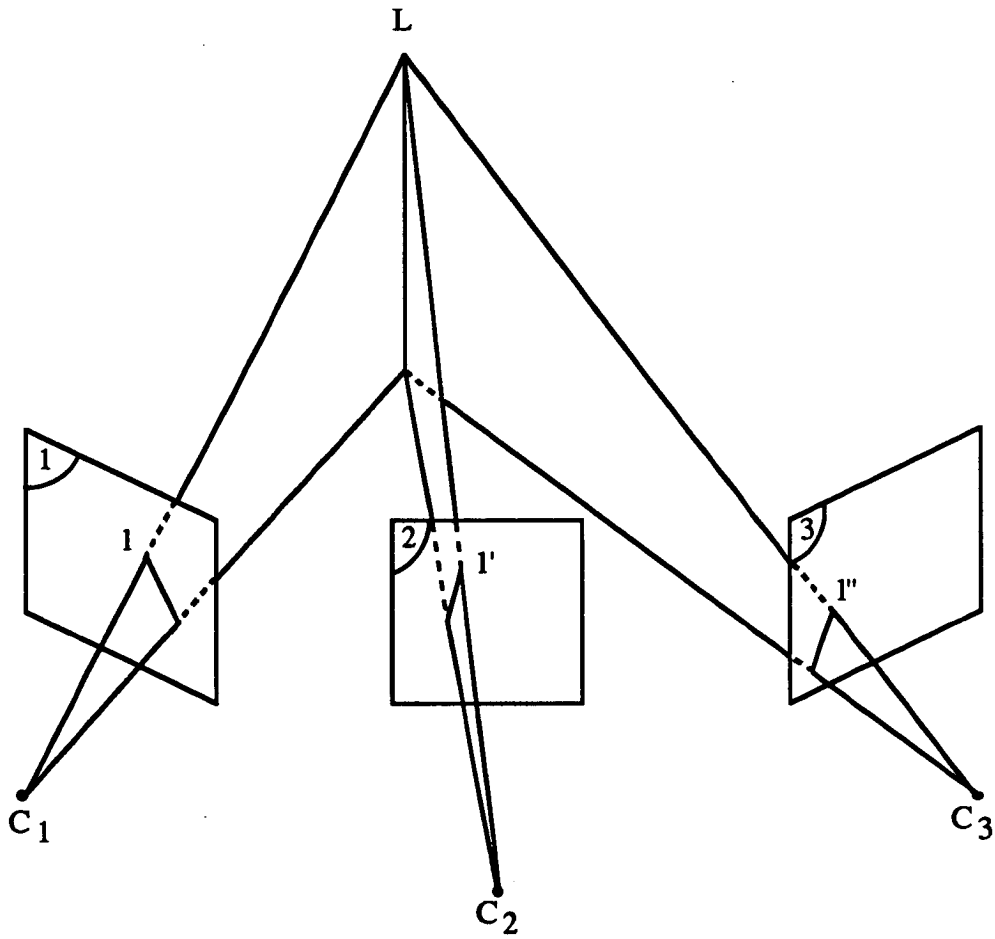


Figure 33: Three lines in three frames

Translation 1 : 50 0 25 (cms)
Translation 2 : 0 100 0 (cms)
Axis of rotation 1 : 1 1 1
Axis of rotation 2 : 0 1 1
Starting point : 0 for all the parameters

1	2	3	4	5	6	7	8	9	10
5	5	0.1	0.61	0.081	0.47	1.5	0.064	0.29	0.44
5	5	0.5	4.4	1.1	3.2	10	0.71	1.8	3.4
5	5	1	12	9.4	7.8	24	2.4	4.9	10
30	5	0.1	0.029	0.16	0.027	0.58	0.059	0.24	0.24
30	5	0.5	0.32	1.2	0.26	5.2	0.33	1.8	2.9
30	5	1	1.1	5.4	0.79	15	1.9	6	10
30	30	0.5	0.39	0.51	0.29	1.1	0.1	1.4	2.5
30	30	1	1.1	1.2	0.82	2.7	1.1	4	7.5
30	30	3	3.8	5.8	2.8	8.9	5.4	23	41
10	20	0.5	2.4	0.61	1.7	2.7	0.74	1.7	3.3
10	20	1	6.1	2	4.2	6.4	2.4	4.6	9.1

1 (resp. 2) : angle rotation 1 (resp. 2) in degrees.
3 : pixel error ϵ (uniform noise in an interval of length 2ϵ).
4 (resp. 5) : relative error on angle 1 (resp. 2) in per cents.
6 (resp. 7) : absolute error on angle of axis 1 (resp. 2) in degrees.
8 (resp. 9) : absolute error on direction of translation 1 (resp. 2) in degrees.
10 : relative error on ratio of norms of translations 1 and 2.

Table 3: Synthetic data

simulated a number of camera motions and measurement noise. The noise is added on the pixels defining the endpoints of the projected segments and varies between 0.1 and 1 pixel. Results are presented in Table 3.

With a 3-camera system commandable in rotation we took 7 stereo triplets of the same scene (one such triplet is shown in Figure 25), each one differing from the previous by a rotation of 5 degrees around a vertical axis. The matches were obtained by using a 3D matching procedure between two views following the trinocular stereo algorithm described in [AL87]. The results appear in Table 4. The deviation of the angle from its predicted value of 5 degrees and of the axis from a perfect verticality being quite stable, we believe that our computations are more accurate than the command of the system. Furthermore,

1	2		3		4	5
0-1-2	206	0.02168	0.996340	-0.085454	5.2	4.9
		0.03186	0.996462	-0.083988	5.4	4.8
1-2-3	158	0.001352	0.997401	-0.072036	5.3	4.1
		0.000696	0.993526	-0.113600	5.2	6.5
2-3-4	108	0.002404	0.995785	-0.091683	5.3	5.3
		0.002502	0.997182	-0.074976	5.3	4.3
3-4-5	63	-0.039143	0.997057	-0.065924	5.3	4.4
		0.002230	0.995984	-0.089499	5.3	5.1
4-5-6	10	0.001226	0.996201	-0.087078	5.2	5
		0.002527	0.997028	-0.077000	5.2	4.4
5-6-7	13	0.014293	0.998154	-0.059027	5.3	3.5
		0.003504	0.995266	-0.097128	5.2	5.6

1: triplet of images used

2: number of matched segments.

3: axis of rotation (first line from image 1 to image 2, second line from image 2 to image 3).

4: angle of rotation.

5: angle of axis relative to the vertical axis of the absolute coordinate system.

Table 4: Real data

it indicates that our method can cope with noisy data.

They somewhat contradict the results reported by Liu and Huang [LH86b,LH86a] who reported a large sensitivity of their method to noise and convergence toward false minima. The Extended Kalman Filtering approach seems to be much more immune to these problems since in all cases our initial guess was the identity matrix for the rotation. Of course, this needs to be tested further. Another big advantage of this method is that it allows us to easily inject any a priori knowledge about the rotation as constraints on the initial covariance matrix $\mathbf{S_0}$.

Remark: A similar remark to the one made in the case of points applies here. Indeed, we could compute the reconstructed 3D line L as a function of the motion D_1 from camera 1 to camera 2, and the motion D_2 from camera 2 to camera 3 (see [FLT87]) for a way of doing this), project it back on the three retinas as l_1, l'_1, and l''_1, and consider the distances $d(l, l_1)$, $d(l', l'_1)$, and $d(l'', l''_1)$. The sum over all matches of these distances is a function of D_1 and D_2 which could be minimized. Again, this has the advantage of weighting less

remote lines than close ones.

2.7.3 Finding the matches

The main difference with the case of stereo is that since the motion of the camera is a priori unknown, it is not possible to use the epipolar constraint to reduce the size of the search space for matches. Other constraints such as continuity (based on the computation of disparity), and ordering (based on epipolar geometry) cannot be used either.

It looks like the matching problem may be very serious. It may not be true for a number of reasons. One such reason is that if the system that extracts tokens and features works fast enough as compared to the image velocity of the tokens, the matching from frame to frame becomes simple. Motion estimation from these matches may be quite inaccurate as pointed out by Nagel [NAG86]. Nonetheless if we track the motion over several frames and wait until the equivalent baseline is long enough, then we may be able to obtain much more robust estimates. A second reason is that in some cases (such as with robots), we may have some good a priori knowledge of the camera motion. This in turns implies that some of the epipolar geometry is known, thus simplifying the matching problem. Nonetheless, we think that this is an area where a lot of useful research could be performed.

2.8 Active 3D Vision

We have seen previously that the problem of matching tokens may be difficult in Stereo and Motion (ambiguity for large disparities or fast motions). Moreover if no contrast is present, then few tokens will be extracted and the 3D maps obtained may be too sparse for the application considered. For these two reasons, systems have been designed which use extra lighting to reduce the matching ambiguity and increase the density of the resulting depth maps.

Examples are the stereo matcher developed by K. Nishihara at MIT [NIS84] where a texture is projected onto the scene to increase the density of features and therefore that of the resulting depth map, or the laser range finders developed at INRIA [FGK*83], MIT [BRO84], and now by a number of companies. Another possibility is the ERIM laser range finder that measures the time of flight of a laser beam to compute distances.

Laser based range finding techniques have the obvious advantages of making the matching in general simpler and increasing the density of data. They have the disadvantage that

they do not work well at depth edges, for surfaces that have a large glossy reflectance component, and that they require an extra source of illumination implying extra power and additional mechanical constraints. Another disadvantage is that lasers emit monochromatic lights, and that therefore the richness of the whole visible spectrum is lost.

Despite these disadvantages, we believe that in a number of constrained industrial environments they are quite enticing and can in general provide another source of 3D information for systems that use passive vision. We think nonetheless that because of their flexibility and simplicity, passive vision systems will be the main sources of 3D data for the robotics systems of the future.

2.9 What needs to be worked on

Here are a number of areas in which we believe considerable progress must and can be achieved:

1. **Classify edges for Stereo and Motion.** A potential source of errors in Stereo and Motion matching in the current algorithms is the small number of stable features attached to the tokens that they use. As we discussed in Section 2.2, it may be possible to reliably extract features that would allow us to classify edges either from the standpoint of their physical origin or from the standpoint of the shape of the intensity function in the vicinity of those edges.

2. **Understand matching errors in Stereo and Motion algorithms.** Indeed, these errors are the source of countless problems when the results are used to build higher level representations than depth maps. On the other hand, no errors seem to be made by the human visual system. The main difference between Computer Vision Stereo algorithms and human perception seems to us to come from the fact that human perception is essentially dynamic: our eyeballs are continuously in motion and so is our head. We believe that this is one key to eliminate errors in Stereo matching by comparing views taken from slightly different viewpoints and comparing them for inconsistencies.

3. **Understand fully the number of solutions for Motion.** In Sections 2.7.1 and 2.7.2 we discussed a number of algorithms to solve for the motion parameters from point and line matches in the case where noise is present. The computation

of the number of solutions of the corresponding equations and the analysis of the degenerate cases have only been partially done and a lot remains to do. We believe this is important in order to understand the next problem.

4. **Understand the errors in Motion estimation as functions of the geometry.** Indeed, when computing motion parameters using for example the algorithms discussed in Sections 2.7.1 and 2.7.2 on synthetic data where we control both the 3D position of the tokens which we project in the image plane, and the amount of noise that we add to these projections, we realize that the error on the motion parameters is a complicated function of the geometry of the problem. Understanding this relation is the key to more robust algorithms.

5. **Find the matches in Motion estimation.** There is still a large potential for good work on the token tracking problem.

6. **Combine Stereo and Motion.** This is a good example of a class of sensory problems where several sensors or sources of information are available and there is need to combine them. The whole theory of sensory integration remains mostly to be done even though some work has recently been devoted to this problem [CRO86,FAF86,AF87,DUR86,SC86]. On the specific problem of integrating Stereo and Motion (for example to correct errors), very little has been done.

3 Shape representation

Now that we have dealt with the problem of obtaining rough 3D data, we would like to use them for a number of application tasks. One such application is the navigation of a mobile system in an unknown environment, another one is the recognition and localization of objects.

Even if these tasks do not have exactly the same requirements in terms of the kinds of representations they use, we believe that they should use a common representation with the following properties:

1. **Object centered so that the effect of rigid motions can be readily assessed.** Indeed, since our representations are intended to support tasks such as recognition from various viewpoints, localization, navigation, manipulation, they should behave simply with respect to the group of 3D rigid motions.
2. **Volume and surface oriented.** In Robotics applications, we are both interested in a volume representation of free space and in a surface representation of the objects and obstacles present.
3. **Work for both sparse and dense data.** We would like our representations to have good convergence properties when the density of measurements increases.
4. **Work for both dense and sparse data.** If many measurements are available, and if the geometry of the environment is simple, we would like to be have the possibility of simplifying the representations.
5. **Easy update when new data becomes available.** We would like the representations to be incremental in the sense that adding new measurements does not imply to recompute everything from scratch.
6. **Efficient computation.** This is connected to requirement 5) but not equivalent to it.

A representation satisfying these requirements having been built, it can then be used to construct other representations more tailored to specific applications. We concentrate first on this representation and propose a fairly general scheme for building it.

3.1 Interpolation through the Delaunay triangulation

In a number of publications, Boissonnat [BOI84,BOI85,BOI86] has proposed roughly to use the Delaunay triangulation to interpolate 3D data obtained for example from a laser range finder. Due to lack of space, we cannot describe here the Delaunay triangulation and refer the interested reader to [BOW81,PS85,WAT81].

We describe here the application of this technique to the interpolation of Stereo data, more precisely to the construction, from the output of the stereo matchers discussed in Sections 2.4, 2.5, and 2.6 which produce line segments in 3D space, of a polyhedral approximation of the objects and obstacles present in the scene and of a volumic representation of free space.

Figure 34 shows a two-dimensional example of the basic principle of the method. Point P represents one of the cameras optical centers, points A, B, C, D, M, E, F, and G are points in a set S, measured by the stereo process. We have computed the Delaunay triangulation of the points in S and we are considering point M. This point is by definition visible from P and therefore all triangles in the Delaunay triangulation intersected by the segment PM should be marked as empty space. These are the textured triangles in Figure 34. By considering more points and the second camera optical center, more triangles can be marked as empty space.

In three dimensions, things are very similar except for the fact that we are now dealing with line segments in 3D rather than points. We assume that the Delaunay triangulation is built for the set S of the segments endpoints.

As shown in Figure 35, for a given stereo segment AB, the visibility property means that all tetrahedra intersected by triangle PAB can be marked as empty space. For example,the triangle PAB intersects tetrahedron $QRST$, the intersection being triangle HIJ. Therefore, tetrahedron $QRST$ can be marked as empty space.

The next question then is, given the Delaunay triangulation of the endpoints of the stereo segments, how can we efficiently detect those tetrahedra which are intersected by at least one triangle PAB, where P is one of the cameras optical center and AB one of the stereo segments. The obvious way of going is by performing the following algorithm:

For all stereo segments AB

  For all unmarked tetrahedra T

    Determine if the intersection of triangle PAB with tetrahedron T is not empty

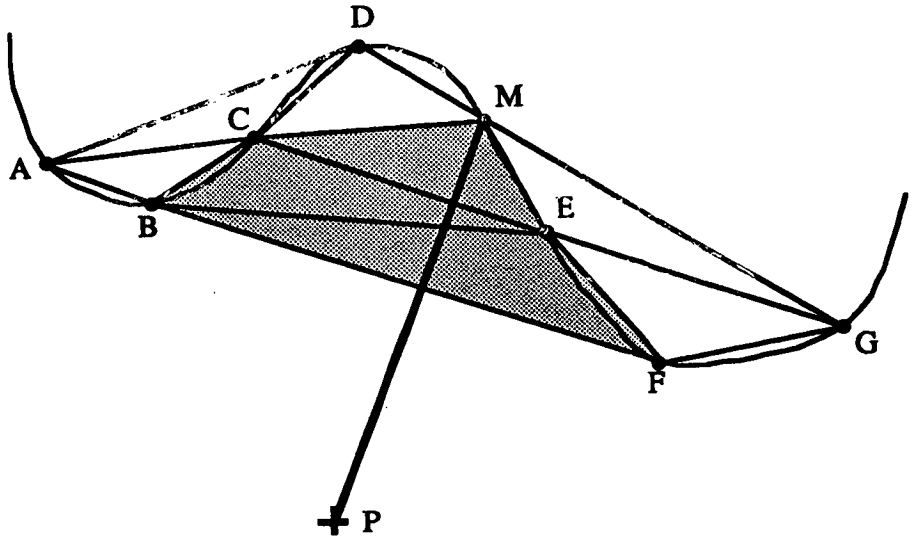


Figure 34: Basic idea in 2D for volume and surface representation

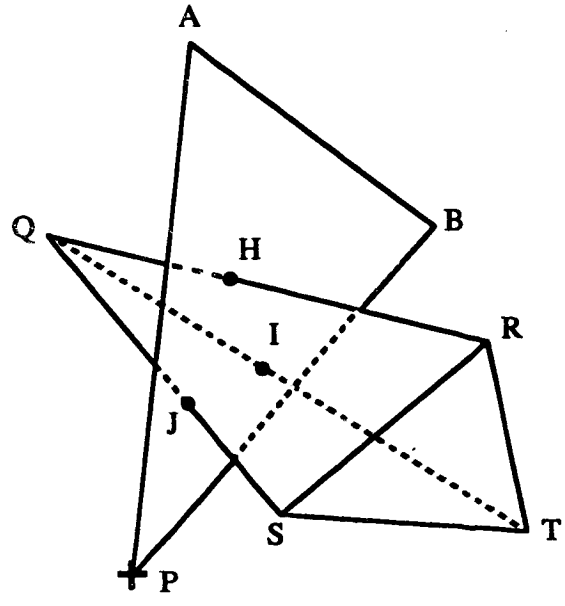


Figure 35: Same idea in 3D

If it is not empty then mark tetrahedron T

Clearly, the computation time of this algorithm depends heavily on the number of intersections which are going to be computed. In order to minimize this number, there are several ways of going which all imply some preprocessing of the data. The obvious idea is to use bucketing techniques, not unlike what was done in Section 2.4.1. For details, the interested reader is referred to [BFLB87].

Assuming that empty tetrahedra have been marked, we can identify two difficulties with this approach. The first one is that since the output of the stereo algorithm is a set of line segments in 3D space, there is no guarantee that in the Delaunay triangulation of their endpoints, the segments will be Delaunay edges. This may be a problem if we wish the reconstructed surface to be a good approximation of the real one.We show next that this can be guaranteed if we allow ourselves to add more points on some of the segments.

3.1.1 Constrained Delaunay Triangulation

Indeed, one problem with the Delaunay triangulation is that it is not immediatly obvious how to modify it so that it is possible to include line segments, triangles or even tetrahedra while making sure that these primitives will be part of the final triangulation. This is a problem known in Computational Geometry as the constrained triangulation problem. A paper by Lee and Lin [LL86], defines a constrained Delaunay triangulation for any planar straight line graph that can be computed in $O(n^2 logn)$ time. We present here an algorithm that runs in $O(n^2)$ in the 2-dimensional case worst-case, can be generalized to any dimension and is very efficient in practice.

We modify the input data by adding more points, in such a way that the resulting Delaunay triangulation is guaranteed to contain, say a set of initial edges. Indeed, from the definition of the Delaunay triangulation, a given set of line segments will be Delaunay edges of the Delaunay triangulation of their endpoints if the discs having those segments as diameters contain in their interiors no other points than those of the segment.

Let S be such a disc attached to a segment s, and $s_1, s_2, \cdots, s_n$ be the segments intersecting that disc. It is possible to partition s in a finite number of subsegments such that none of the discs attached to those segments intersect any of the s_i's. Indeed, let d be the smallest distance of the s_i's to segment s. The way to compute the distance of a segment to another segment is shown in Figure 36.

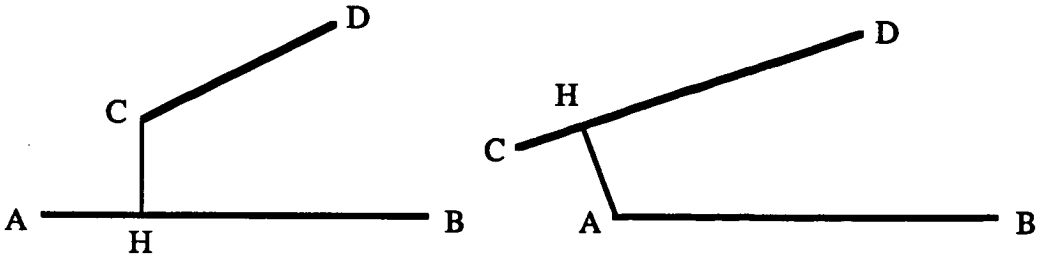


Figure 36: Computing distances between segments

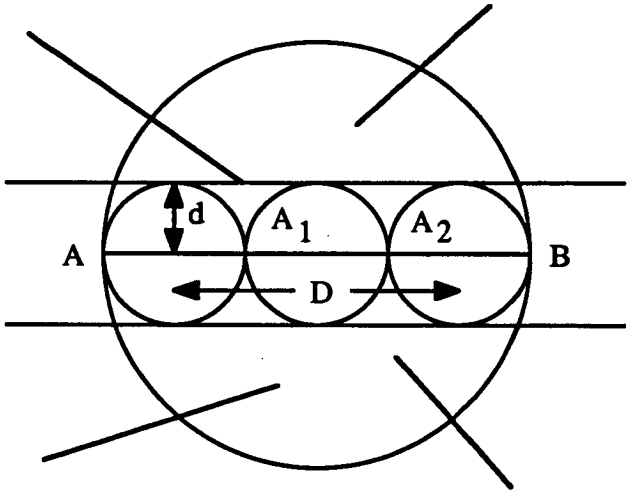


Figure 37: Adding new points on the segments to make them Delaunay edges

Assuming that segments do not cross, d is stricly positive. If D is the length of s, we can divide s into $\lceil D/2d \rceil$ intervals of length less or equal $2d$. This corresponds to adding $\lceil D/2d \rceil - 1$ points, the endpoints of these segments. Therefore, as shown in Figure 37, the discs having those segments as diameters do not contain any points on other segments. In this Figure, points A_1 and A_2 have to be added to the original set of points.

In the case where AB is connected to one segment BC, we consider the plane perpendicular to AB in B. If BC is on the other side of this plane than AB, then we can ignore BC (in fact, we can ignore all the line segments on the same side of the plane) and the previous technique for splitting AB can be used (see Figure 38).

When they are on the same side, there exists a sphere passing through point B, a point A_1 of AB, and a point C_1 of BC, and not intersecting any other segment. This is shown

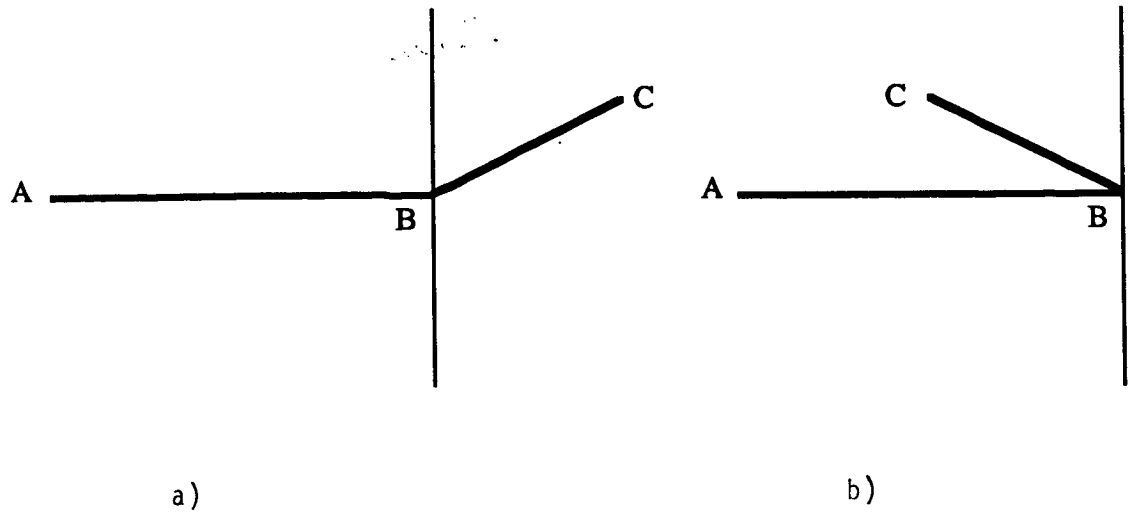


Figure 38: Segments connected to AB: a) Do not cause problems. b) Require special processing.

in Figure 39. Points A_1 and C_1 have to be added to the original set, and the previous technique can then be applied to segments AA_1 and CC_1, if necessary.

After this preprocessing, the resulting augmented set of line segments satisfies the condition that they are Delaunay edges of the Delaunay triangulation of their endpoints. This preprocessing has been described in the 2-dimensional case; it can be extended to any dimension in a straightforward manner.

An interesting question at this point is how many points do we add in this process ? the answer is that the worst case is $O(n)$. Indeed, it can be seen from Figure 37 that for a given segment AB of length D, there exists a number $d > 0$ such that in the band parallel to AB of width $2d$ there are no other points belonging to the other segments. We then only need to add $\lceil D/2d \rceil - 1$ points to segment AB so that the corresponding segments thus created satisfy the constraint.

We present now the algorithm, based on the above ideas, that computes a Delaunay triangulation constrained to contain a set of edges E:

Compute the Delaunay Triangulation of the end-points of the edges in E
For any edge, say AB, in E that is not an edge of the triangulation do:

Find $i \in [1, n]$ such that the distance d_i between s_i and AB is minimal;

Add the corresponding new points (as described above) in a list of new points;
Update the triangulation by introducing the new points.

The computation time of this algorithm is $O(n^2)$ in the planar case and $O(n^3)$ in the 3D case, if we use the sequential algorithm of the Delaunay Triangulation. Let us do the

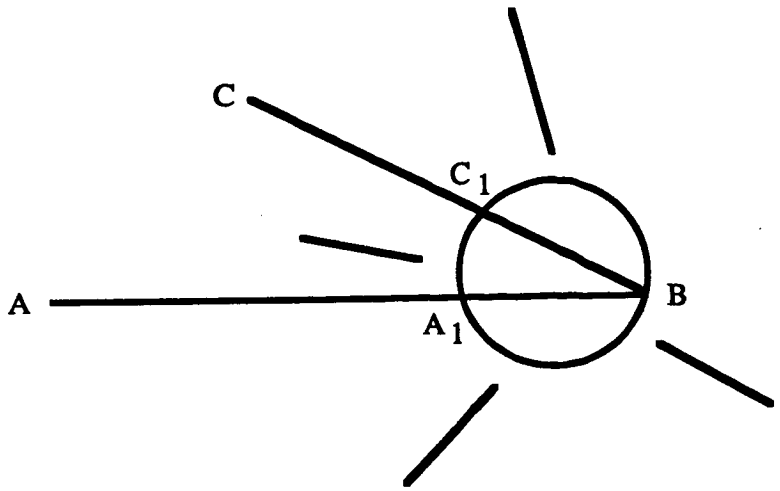


Figure 39: Special processing for connected segments

analysis for the 3D case. The complexity comes from the Delaunay triangulation which is $O(n^3)$ worst-case and from step a). Since we may have $O(n^2)$ tetrahedra in the worst-case in the triangulation, we may have to compare AB with $O(n^2)$ edges thus adding another $O(n^3)$ term to the complexity. In practice, computation of the Delaunay triangulation by the iterative technique is linear. In order to reduce the second source of complexity, we use *bucketing techniques.* Indeed, we do not need to compare for each edge AB, the distance between AB and each other segment. The segments which must be considered are those intersecting the sphere whose diameter is the segment AB.

The bucketing techniques consists here in dividing the space into cubic or parallelepipedic buckets and to compute for the sphere the buckets which cover it and for each segment the buckets intersected.

We then compute the distances between the segment AB and each segment which intersects the sphere, i.e. the segments which have at least a bucket in common with the sphere. In practice, this makes the complexity of step a) linear at the cost of preprocessing the data to compute the buckets.

Figure 40 shows an example of a 3D office scene, Figure 41 shows a cross-section by a horizontal plane of the original constrained triangulation before marking tetrahedra, and Figure 42 shows the same cross-section of the remaining tetrahedra after exploiting the visibility property.

Figure 40: 3D office scene

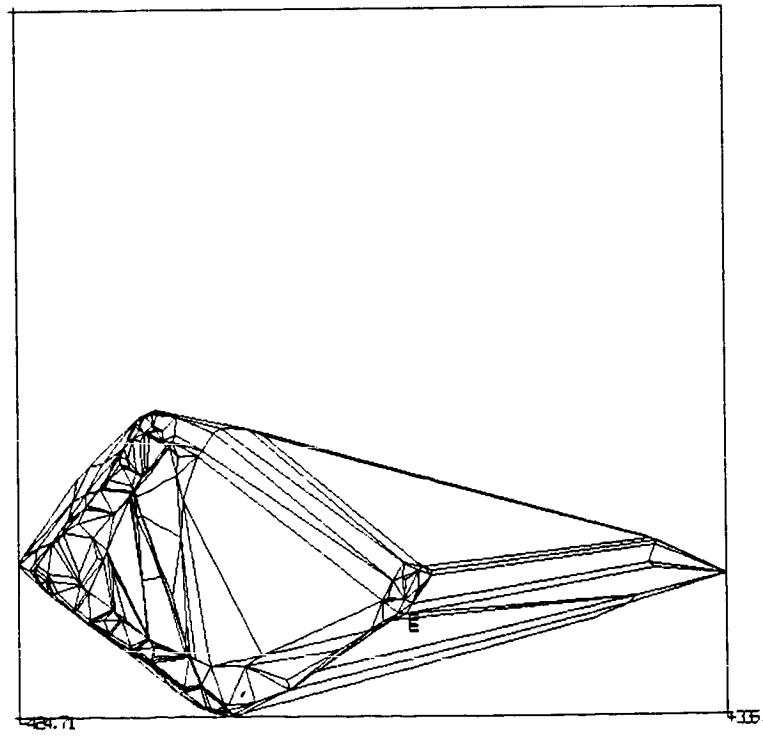


Figure 41: Cross-Section by a horizontal plane of the original constrained Delaunay triangulation

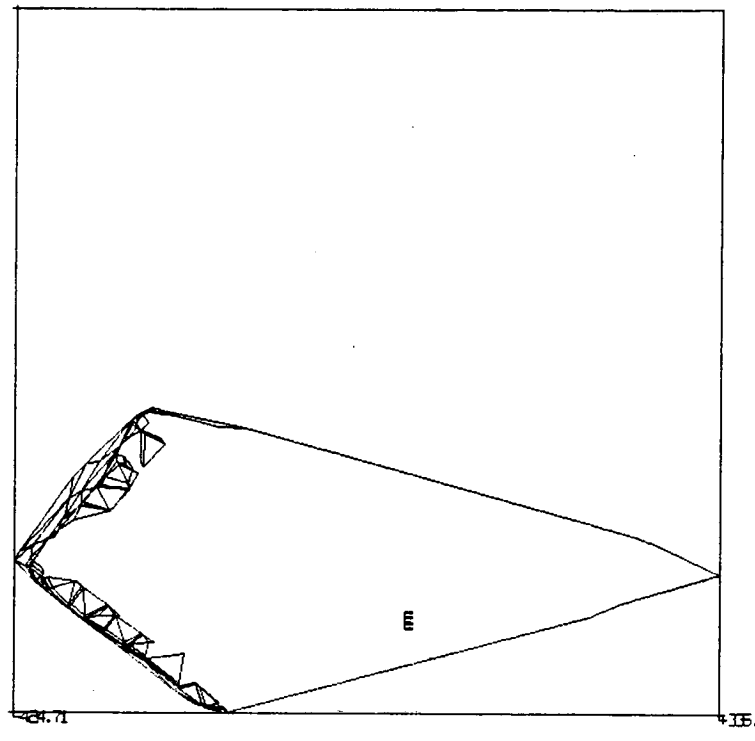


Figure 42: Cross-Section by a horizontal plane of the remaining tetrahedra after using the visibility constraint

3.2 Finding important curves on a surface

The methods we describe next apply more to the case where the previous triangulation has been built from a dense range map such as the one provided by a laser range finder. We draw heavily on the work of Brady and his colleagues [BPYA85,PB85].

Curves on a surface are a useful source of constraints on the geometry of the surface and can be used as a basis for its description. There are three types of curves we are interested in detecting and describing:

1. Bounding contours
2. Lines of curvature, geodesics, and parabolic lines
3. Curves where significant surface changes occur

Bounding contours are either extremal, i.e places where the surface normal turns smoothly away from the viewer, or mark discontinuities of some order of the surface.

3.2.1 Bounding contours

For extremal boundary contours, there exists an important relationship between the curvature of the projection on the retina of the boundary curve and the Gaussian curvature of the surface at points along the boundary curve. This relationship is the following [KOE84,BPYA85]:

> *the sign of the gaussian curvature of the surface at points along the boundary curve is the same as the sign of the curvature of the projection of the boundary curve*

The proof of this result is a little bit involved and we refer the interested reader to the previous references.

3.2.2 Lines of curvature, geodesics

Lines of curvature give the locally flattest coordinate system intrinsic to the surface. They are local statements about the surface. In practice, we aim at describing surfaces at a larger scale. This is done by linking the local directions to form smooth curves. As pointed out in [BPYA85], there are generally infinitely many such curves and we want to make explicit in our representation only a small finite subset of those.

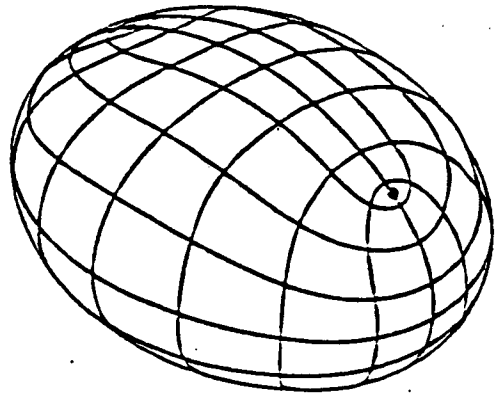


Figure 43: Lines of curvature for an ellipsoid

Ways of obtaining such a subset is by adding constraints to the curves. Possible constraints are:

1. the curve is planar

2. the descriptors along the curve are constant

Indeed, as it is shown in Figure 43, there are only three planar lines of curvature for an ellipsoid, and they are the intersections of the surface with its planes of symmetry. The surface is effectively described by these curves. Brady [BPYA85] has developed a program that, given a dense depth map of an object, smoothes the depth map by gaussian filtering, and computes at every point the principal directions and the surface curvatures. Lines of curvature can be found by linking the principal directions at the individual surface points. Results are shown in Figure 44 for an oil bottle and a coffee mug.

Another set of potential good candidates is the set of geodesics. However, potential disadvantages of using the geodesics as descriptors are that there are too many of them (one through every point on a surface in every direction) and that if we add the constraint that the geodesics be planar, then they are lines of curvature.

3.2.3 Finding surface discontinuities

Another potentially very attractive way of describing a surface is to detect the places where it changes abruptly. This is very similar to the edge detection problem with one very important difference. The difference is that edge detectors usually operate on intensity images where the measured intensity is a mixture of object surface geometry, object surface

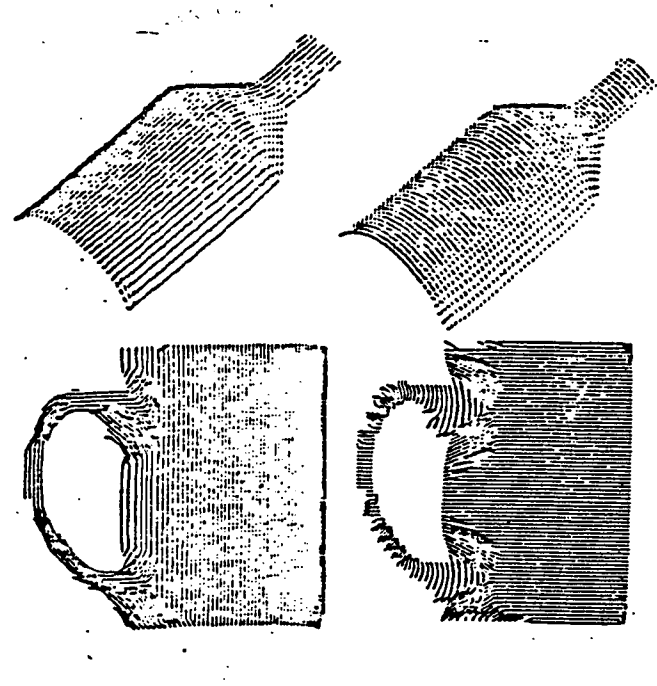


Figure 44: Results of linking the principal directions on the surface of an oil bottle and a coffee mug ([PB85])

reflectance, and illumination . The goal of Image Analysis is to extract from such an image a set of intrinsic features which characterize the geometry of the objects in the scene. Examples of such features include surface normals, lines of curvature, discontinuities in depth, in the normals, etc $\cdots$.

We are facing here a very different situation, and apparently a simpler one, in the sense that , unlike the case of an intensity image, range is available to us. Therefore edge models are really surface models and we do not have such things as reflectance, texture, shadow edges.

In order to detect the surface changes we need to characterize the type of changes we are interested in, that is we must define quantitative models for them. We must also propose algorithms that reliably detect those models in range data, taking into account the inevitable presence of noise. Ponce and Brady [PB85] consider three types of models:

- steps where the surface height function is discontinuous
- roofs where the surface is continuous but its normal is discontinuous
- shoulders, which consist of two roofs and correspond to a step viewed obliquely.

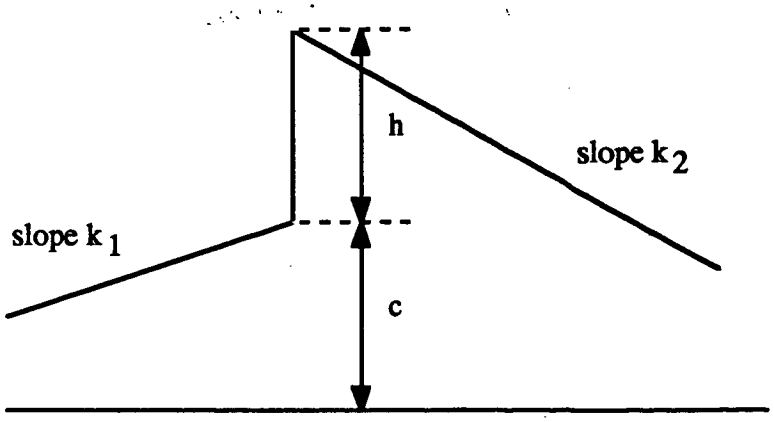


Figure 45: Step model

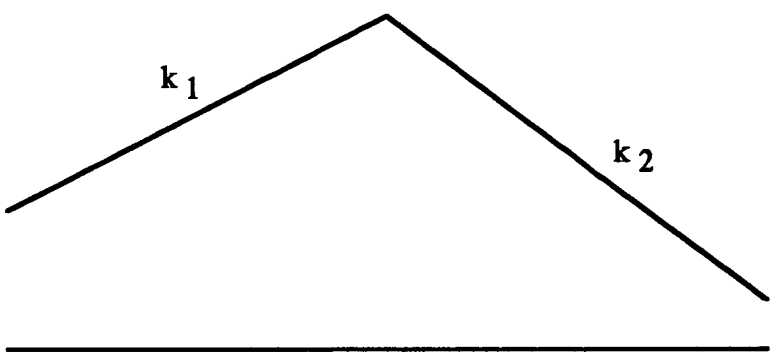


Figure 46: Roof model

The three models are shown in Figures 45- 46. They are one dimensional models and therefore the surface is modelled locally as cylindrical.

The basic idea is to smooth the surface data, using gaussian smoothing, at different scales, extract significant features and use them to **identify** the presence of the models, track these features at different scales to **localize** the identified model on the surface. It turns out that roofs consist of extrema of the dominant curvature, on the other hand, steps and shoulders consist of parabolic points, that is zero crossings of the Gaussian curvature and can be distinguished by their behavior at different scales.

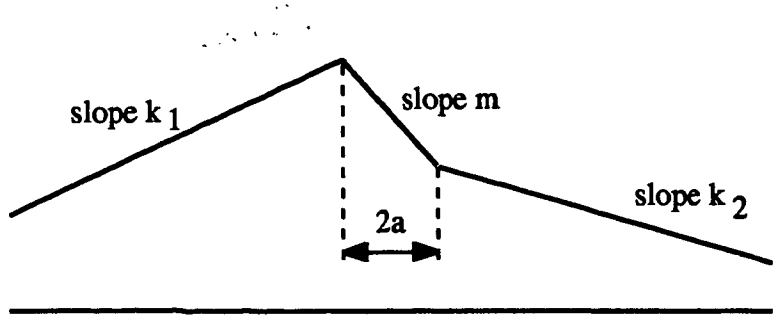


Figure 47: Shoulder model

The key to using the combination of cylindrical models and detection of lines of curvature is in the following theorem [BPYA85]:

> *The convolution of a cylindrical surface with a Gaussian distribution is cylindrical*

Since we also know that the lines of curvature of a cylinder are in the plane of the curve and parallel to the generating line then we know that this property is conserved after Gaussian filtering, thus allowing us to detect the models by computing the derivatives of the principal curvatures in the direction of corresponding line of curvature.

For a planar curve given by its equation $y = f(x)$, the curvature is given by the formula:

$$\kappa(x) = \frac{f''}{(1 + f'^2)^{3/2}} \tag{11}$$

From formula 11 and the equations describing the step model, it is easy to derive the fact that for a step smoothed by a gaussian of variance σ, the curvature κ has a zero crossing at the point $x_\sigma = \sigma^2\delta/h$ (h is defined in Figure 45 and $\delta = k_2 - k_1$). Also, using the fact that the second derivative of f_σ (f smoothed by a gaussian of variance σ^2) is 0 at x_σ, it is also easy to show that:

$$\frac{\kappa''(x_\sigma)}{\kappa'(x_\sigma)} = -2\delta/h$$

stating that the ratio of the second and first derivatives of the curvature at the zero crossing is constant over the scales.

For the roof model, we can specialize the previous analysis to the case where $h = 0$ and show that the curvature satisfies the following relation:

$$\kappa(x, \mu\sigma) = \kappa(x/\mu, \sigma)$$

From this, it follows that the extremum value of κ is proportional to $1/\sigma$, and that its distance from the origin is proportional to σ.

Finally, for the shoulder model, it can be shown that there exists a zero crossing of the curvature at the point $x_\sigma = (\sigma^2/2a)log(\delta_1/\delta_2)$. It is then straightforward to show that:

$$\frac{\kappa''(x_\sigma)}{\kappa'(x_\sigma)} = -\frac{1}{a}log(\frac{\delta_1}{\delta_2})$$

so the ratio of the first and second derivatives at the zero crossing is constant over scale.

These three models can now be used to detect and localize surface intersections by tracking extrema of curvatures (roofs) and parabolic points (steps and shoulders). The technique goes as follows.

- First smooth the original depth map with a gaussian distribution at a variety of scales σ.
- Second, compute at each scale the principal curvatures and their directions, as well as the first and second derivatives of the principal curvatures in their associated directions.
- Third, mark parabolic points (zero crossings of the gaussian curvature) and (directional) maxima (minima) of the maximum (minimum) curvature.
- Fourth, track the extrema and zero crossings across scales, from coarse to fine.The result is a forest of points, a "fingerprint" [YP83a,YP83b]. A path in the forest corresponds to a feature point.
- Fifth, eliminate points at the finest scale which have no ancestor at the coarsest scale on their path (detection). The remaining points are located from the finest scale.
- Sixth, label each remaining path as roof, step, or shoulder, according to the behavior of its curvature and its first and second derivatives across scales, using the previous characteristic relations. Paths which do not follow the properties of any of those models are eliminated.

Examples and results are shown in Figure 48 taken from Ponce and Brady [PB85].

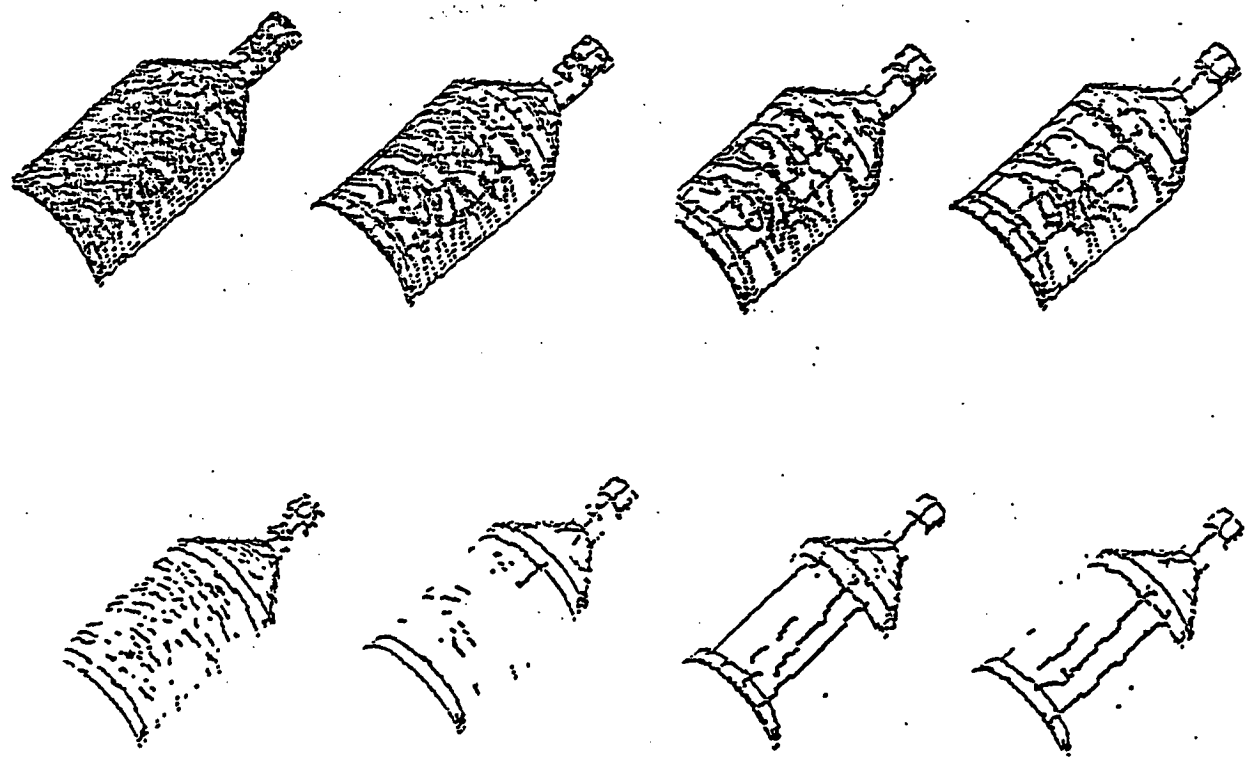


Figure 48: Detection of discontinuities on the surface of an oil bottle ([PB85])

3.3 Finding surface patches

So far, we have seen how to represent a set of measured points by finding a simplicial polyhedron that interpolates these points. When the density of the points is high, this polyhedron is a very fine approximation of the set of points which may not be suitable for many applications. We have seen in Section 3.2 how to use this structure to make explicit important curves, mostly from the computation of lines of curvature.

In this Section, we want to develop a complementary approach to that of finding curves, namely that of finding surface patches. The kind of surface primitives that can be used is highly constrained by their accessibility from the raw data. More precisely, we want them to be:

1. robust to measurement noise

2. viewer independent

Condition 1) implies that we cannot use high-degree polynomials since they will tend to "follow" the noise. Condition 2) implies that if we approximate the points in two different coordinate systems, then the approximations should relate to each other by the same

transformation as that which relates the two coordinate systems. These considerations have prompted us to use first and second degree polynomials (planes and quadrics) to approximate surface patches.

3.3.1 Representing planes and quadrics

A plane is represented by its unit normal vector $\mathbf{n}$ and its distance to the origin d (see Section 2.5.2). The plane equation is given by:

$$\mathbf{n}^T\mathbf{x} = d \qquad \text{where} \qquad \mathbf{x} = [x, y, z]^T$$

This implies that a plane has two equivalent representations $(\mathbf{n}, d)$ and $(-\mathbf{n}, -d)$.

A standard representation for a quadric is a symmetric 3×3 matrix $\mathbf{A}$, a 3×1 vector $\mathbf{v}$ and a number d, the equation of the quadric is then given by:

$$\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{v} + d = 0$$

Notice that the representation $(\mathbf{A}, \mathbf{v}, d)$ is defined up to a scale factor; we show in the next Section how to impose a constraint on matrix $\mathbf{A}$ to reduce the ambiguity of the representation to two.

3.3.2 Surface fitting

When constructing the regions of the object which are best approximated by planes or quadrics, we have to solve repeatedly the following problem. Given a set of points forming a region, find the plane (quadric) that minizes the error measure:

$$E = \sum d^2(M_i, S) \tag{12}$$

where the M_i's are the points in the region and d is the distance of M_i to the surface S. In the case of a plane $(\mathbf{n}, d)$, it is well known that $d(M_i, P) = \mathbf{n}^T\mathbf{x}_i - d$. In the case of a quadric Q, there is no such simple formula relating the representation of Q and the distance d. In fact, it can be shown, that for a given point M, its distance to a quadric is found by solving an equation of degree 3. For the sake of simplicity, and in analogy to the planar case, we define as the point error measure the value:

$$d^2(M_i, Q) = \mathbf{x}_i^T\mathbf{A}\mathbf{x}_i + \mathbf{x}_i^T\mathbf{v} + d$$

In the case of a quadric with a center, this is equal to the square of the euclidian distance between M_i and m_i, the intersection of the line CM_i (C the center of the quadric) with the quadric.

The problem is now to minimize criterion E in the case of a plane and a quadric. The case of a plane is classic [DH73]: the normal to the best plane is the eigenvector of length one, $\mathbf{n}_{min}$ of the covariance matrix $\mathbf{C}$ of the points in the region, associated with the smallest eigenvalue λ which is also the minimum error E_{min}. The covariance matrix is given by:

$$\mathbf{C} = \sum \mathbf{A}_i \mathbf{A}_i^T, \qquad \mathbf{A}_i = \mathbf{x}_i - \mathbf{x}_g \qquad \text{and} \qquad \mathbf{x}_g = 1/N \sum \mathbf{x}_i$$

The distance to the best approximating plane is given by:

$$d_{min} = -1/N \sum \mathbf{n}_{min}^T \mathbf{x}_i$$

In the case of a quadric, we represent it, for the purpose of approximation, by ten numbers $a_{1\cdots10}$ related to the previous description by the following relationships:

$$\mathbf{A} = \begin{bmatrix} a_1 & a_4/\sqrt{2} & a_5/\sqrt{2} \\ a_4/\sqrt{2} & a_2 & a_6/\sqrt{2} \\ a_5/\sqrt{2} & a_6/\sqrt{2} & a_3 \end{bmatrix}$$

$$\begin{aligned} \mathbf{v} &= [a_7, a_8, a_9]^T \\ d &= a_{10} \end{aligned}$$

As we noted earlier, the representation for a quadric is not scale invariant, and therefore so is criterion E. This implies that if we do not add a constraint on the a_i's, the minimum of E is 0, attained for $a_i = 0$ for all i's. Since there does not exist a "natural" constraint like $\|\mathbf{n}\| = 1$ in the case of a quadric, several constraints can be considered. Among those proposed in the literature:

1. $\sum a_i^2 = 1$
2. $a_{10} = 1$
3. $Tr(\mathbf{A}\mathbf{A}^T) = \sum a_i^2 = 1$

we select the third one since it is the only one invariant with respect to rotations and translations. This implies that if we use 1) or 2), the solution of the minimization of E in two different coordinate systems will not be the same quadric!

Let us define the vector $\mathbf{p} = [a_1, \cdots, a_{10}]^T$. Criterion E can be rewritten as:

$$E = \sum \mathbf{p}^T \mathbf{M}_i^T \mathbf{M}_i \mathbf{p} = \mathbf{p}^T \mathbf{M} \mathbf{p} \quad \text{where} \quad \mathbf{M} \quad \text{is a} \quad 10 \times 10 \quad \text{symmetric matrix}$$

and $\mathbf{M}_i = [x_i^2, y_i^2, z_i^2, x_i y_i, x_i z_i, y_i z_i, x_i, y_i, z_i, 1]^T$. The constraint 3) can be written as:

$$\|\mathbf{B}\mathbf{p}\|^2 = 1$$

where $\mathbf{B}$ is a 6×10 matrix which "selects" the first six elements of vector $\mathbf{p}$. The solution to the optimization problem can be found in [FH86].

3.3.3 Region growing

In the Computer Vision literature there are traditionnally two ways of segmenting a picture, a curve, an object [PAV78]. The first method called the split technique starts from the entire object, let us say, checks if it is homogeneous according to some criterion, and if not, splits it into a number of sub-objects. The process is then applied recursively until either the objects become too small or homogeneous. The major question is where to split and has not received any satisfying answer in more than one dimensions. So far, the only splitting schemes have been defined with respect to a fixed coordinate system, usually the discrete grid on which the object is defined, like in quadtrees or octtrees. This is quite disrupting since we insist on having representations intrinsic to the object rather than on its frame of reference.

In this part, we focus on a region growing paradigm that starts from the simplicial polyhedron obtained, for example by the techniques of Section 3.1, and starts merging the initial triangular regions into larger ones best approximated either by planes or by quadrics. The process stops when either only one region is left or when the total approximation error, as measured by criterion E above, reaches some tolerance threshold. The scheme can easily incorporate the information provided by the results of Section 3.2 about curves on the surface where significant changes occur. This is simply achieved by forbidding two regions to merge if their union contains in its interior one of the previous curves.

The merge procedure works then as follows. At every iteration, the pair of regions R_i and R_j which produce the smallest error $E(R_i \cup R_j)$ and such that $R_i \cup R_j$ does not contain any of the lines computed in Section 3.2 are merged. This guarantees that the total error E grows as slowly as possible and can be easily implemented by maintaining a heap of all pairs of adjacent regions [AHU74], the best pair is then always at the top of the heap. The merge consists in updating the region adjacency graph (RAG) and the heap. The algorithm can be described as follows:

INPUT: a simplicial polyhedron interpolating the data point a tolerance T on the maximum approximation error

OUTPUT: a segmentation of the points into regions best approximated by a plane or a quadric

Initialization: Build the region adjacency graph of the set of initial triangles.
Initialize the approximation error for each region to 0.
For each pair of adjacent regions, compute the error for the best approximating plane (quadric).
Use these values to build a heap such that the top of the heap is the pair of adjacent regions R_i, R_j with the smallest error E_{ij}.

Iteration: Update the total error: $E^n = E^{n-1} + E_{ij} - E_i - E_j$
If E^n is less than the tolerance T

then
remove the top of the heap and update it (this involves recomputing the errors E_{ik} and E_{jl} for all regions R_k and R_l adjacent to R_i or R_j and reinserting the corresponding pairs in the heap)
Update the RAG

else $STOP$

A few points need to be clarified in this algorithm. When we compute the best approximating plane (quadric) for the union of two adjacent regions R_i and R_j, we consider a plane or a quadric only if both R_i and R_j are currently approximated by a plane and only a quadric in the other cases. In the first case, the type of surface that yields the smallest error is selected. Also, when using the curves of discontinuity obtained in Section 3.2, we

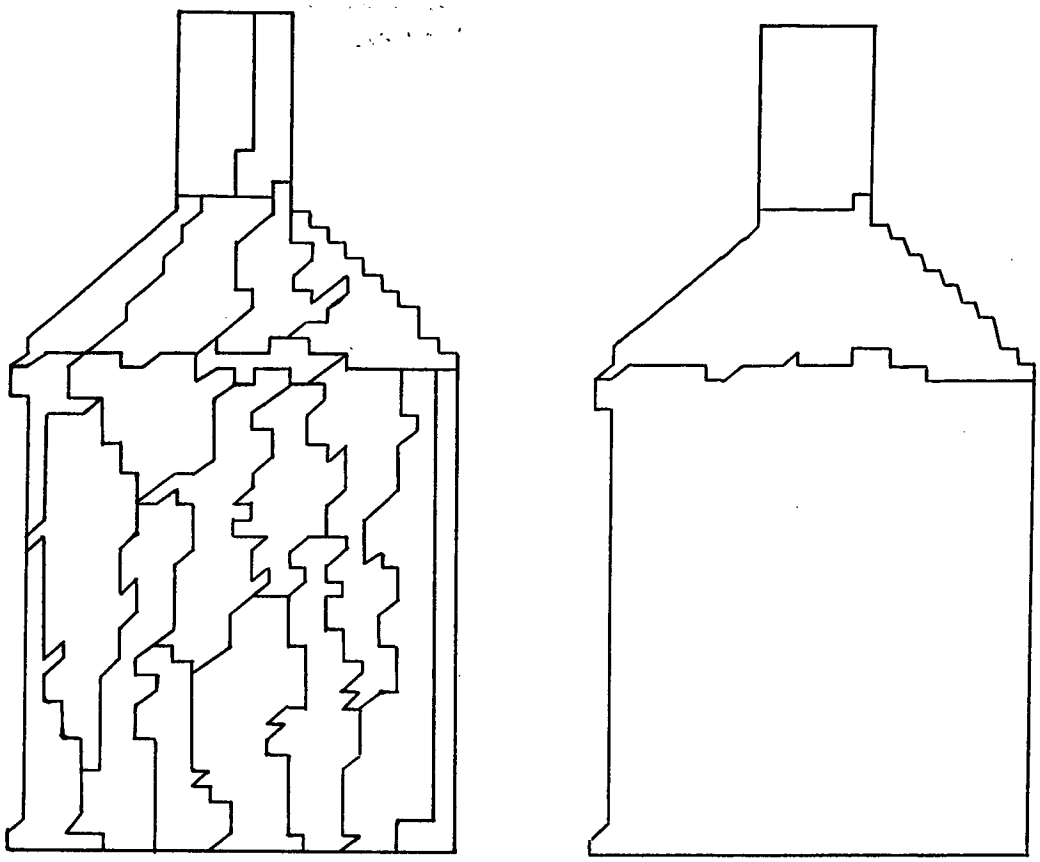


Figure 49: Segmentation of the surface of the oil bottle: a) Using only planar patches. b) Using quadric patches.

do not insert in the heap pairs of regions such that their union contains part of one of these curves. Results of this segmentation program are shown on the oil bottle in Figure 49.

3.4 What needs to be worked on

The question of shape representation cannot be separated from the applications. We focus in this paper (see next Section) on applications such as recognition and localization of rigid objects. The kinds of representations which are needed for these applications may be different from those needed for manipulating objects, or building them, or designing them for specific tasks. And this is only for manufactured objects, but if we extend our realm to the class of natural objects such as rocks, rivers, flames or even living beings such as plants, trees, animals and humans, then it is likely that the kind of models needed will be very different. In particular they should incorporate the notion of growth, of dynamic evolution [THO61,THO72,ARN86]. This delimits a very large research territory which is

mostly unexplored and where we believe that some of the future efforts should be invested in the long term. Closer to our everyday preoccupations, we think that the following areas should be considered:

1. the question of symmetry is important in general for many objects and needs more attention even though some preliminary attempts have been made at understanding the problem [BA84,TWK87].
2. the relationships between Computational Geometry [PS85] and shape representation should also be investigated further, in connection with other issues discussed in Section 3.4.
3. the relationship between shape and function is extremely important if we want to be able to design flexible Vision systems. Very little is known on the subject, and a lot more needs to be done [WBKL84,CB87].

4 Shape recognition/localization

4.1 Position of the problem

We have seen in the previous Sections a number of ways for constructing 3D representations of objects in terms of geometric primitives. We have studied ways of reliably extracting these primitives from a number of sensors outputs and we would like now to address the question of how to use these representations for recognizing and localizing objects or for the navigation of a robot. In this Section we concentrate on the first of these two applications, referring the reader interested in the second to [AF87].

There have been many solutions proposed in the last twenty years or so for solving this problem in a various number of different context and the difficulty of the problem is a function of many variables. We present here a number of these variables:

- First, objects can be one-, two-, or three-dimensional. Examples of one-dimensional objects are waveforms of various kinds, examples of two-dimensional objects are flat parts such as keys, coins, etc $\cdots$ for which the third dimension is negligible with respect to the other two. Examples of three dimensional objects can be found by looking around.

- Second, objects can be completely visible (observed) or partially hidden. It is almost always the case in robotics applications that the objects to be dealt with are partially hidden. The fact that parts of the object may be missing seriously adds to the difficulty of the recognition problem.

- Third, objects can be rigid or not. Non rigid objects range from articulated bodies such as a robot manipulator to continuously deformable objects such as clouds.

- Fourth, and related to this distinction between rigid and non-rigid objects is the distinction between recognizing one instance of a specific shape such as a given chair of known shape and dimensions and recognizing classes of objects such as the class of chairs for example. An articulated object is somewhat in-between in the sense that there exists an analytical parametrization of all its possible configurations whereas very likely such a parametrization does not exist for the class of chairs.

- Fifth is the question of noise. Typically, measurements made in the scene (and

sometimes also when building the model) are noisy. The amount of noise makes the problem more or less difficult.

In this Section we concentrate exclusively on the problem of recognizing and localizing specific instances of partially overlapping rigid 3D objects. We assume therefore that every object is defined by an accurate model $M = (M_1, \cdots, M_n)$ composed of number of geometric primitives M_i of the types we have described in Section 3, and that a number of visual sensors provide us with a similar description $S = (S_1, \cdots, S_p)$ for the observed scene.

The recognition problem is then to produce a list of pairs of scene/model primitives $\mathcal{R}_n = ((M_1, S_{i1}), \cdots, (M_n, S_{in}))$ where the S_{ij}'s are either scene primitives or the special symbol NIL, indicating that the model primitive M_j is not present in the scene.

The localization problem is to find the best rigid transformation $\mathbf{T}$ such that, when applied to the identified model primitives brings them as close as possible to their corresponding scene primitives. A rigid transformation is the product of a translation and a rotation. We will make heavy use of the rigidity constraint to quickly find the best matches between model and scene primitives.

4.2 Search approach to the problem

The complexity of the recognition problem depends directly on the number of scene (p) and model (n) primitives since the number of sequences $\mathcal{R}$ is n^p. This becomes rapidly extremely large when n and p increase and it is therefore impossible to explore all possibilities. Nonetheless, the brute force approach consisting in exploring the interpretation tree of Figure 50 yields itself very nicely to the implementation of the rigidity constraint. The basic paradigm that we use is that of recognizing while localizing. This can be achieved as follows. For every path from the root in the interpretation tree corresponding to a partial regognition $\mathcal{R}_k = ((M_1, S_{i1}), \cdots, (M_k, S_{ik}))(k < n)$, we compute, using methods to be described later, the best rigid transformation $\mathbf{T}_k$ from model to scene for those primitives. Since the measurements are noisy, the residual error ε_k is non zero. Also, k minus the number of model elements assigned to NIL is a measure g_k of how much of the object has been found in the scene so far. A combination of ε_k and g_k is the cost associated with the partial recognition $\mathcal{R}_k$. We can then apply $\mathbf{T}_k$ to the next unmatched model primitive M_{k+1} and consider only as possible candidates for M_{k+1} those unmatched scene primitives

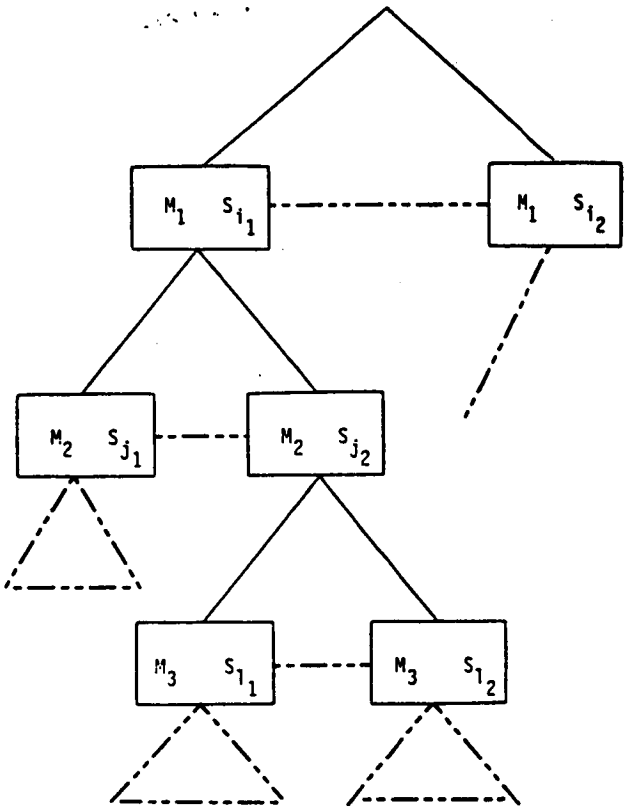


Figure 50: Interpretation tree

which are sufficiently close to $\mathbf{T}_k(M_{k+1})$, thus considerably reducing the breadth of the interpretation tree.

4.3 Localization of 3D objects from 3D measurements

We consider here the localization of 3D objects from 3D data [GL83,BH86,FH86]. Every three-dimensional rigid transformation can be considered in an infinite number of fashions as the product of a translation and a rotation. If we constrain the axis of the rotation to go through a given point and assume that it is applied first, then this decomposition is unique. We denote by $\mathbf{R}$ the orthogonal matrix representing the rotation and by $\mathbf{t}$ the translation vector.

We assume that the geometric primitives that define models and scenes are points, lines, and planes. Points are represented by three-dimensional vectors, planes by their normal $\mathbf{n}$ and their distance to the origin d. Lines are usually defined by two points and we represent them by their Plücker coordinates $(\mathbf{m}, \mathbf{m}')$ as follows. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be the

two points defining the line, then its Plücker coordinates are given by:

$$\mathbf{m} = \mathbf{p}_2 - \mathbf{p}_1$$
$$\mathbf{m}' = \mathbf{p}_1 \wedge \mathbf{p}_2$$

Notice that $\mathbf{m}.\mathbf{m}' = 0$. We have to determine how these representations are transformed through a rigid 3D transformation.

A point M represented by $\mathbf{p} = \mathbf{OM}$ is transformed to $\mathbf{p}' = \mathbf{R}\mathbf{p}+\mathbf{t}$. A line L represented by $(\mathbf{m}_1, \mathbf{m}_2)$ is transformed to $(\mathbf{R}\mathbf{m}_1, \mathbf{R}\mathbf{m}_2 + \mathbf{t} \wedge \mathbf{R}\mathbf{m}_1)$. A plane P represented by $(\mathbf{n}, d)$ is transformed to $(\mathbf{R}\mathbf{n}, d + \mathbf{t}.\mathbf{R}\mathbf{n})$. Therefore, if we match a point $\mathbf{p}'$ in the scene to a point $\mathbf{p}$ in the model, the following relationship must hold:

$$\mathbf{p}' = \mathbf{R}\mathbf{p} + \mathbf{t} \tag{13}$$

If we match a line $(\mathbf{m}'_1, \mathbf{m}'_2)$ in the scene to a line $(\mathbf{m}_1, \mathbf{m}_2)$ in the model, the following relationships must hold:

$$\|\mathbf{m}_1\|\mathbf{m}'_1 = \|\mathbf{m}'_1\|\mathbf{R}\mathbf{m}_1 \tag{14}$$

$$\mathbf{m}'_2 \wedge (\mathbf{R}\mathbf{m}_2 + \mathbf{t} \wedge \mathbf{R}\mathbf{m}_1) = \mathbf{0} \tag{15}$$

Using equation 14, we rewrite equation 15 as:

$$\mathbf{m}'_2 \wedge (\mathbf{R}\mathbf{m}_2 + r\mathbf{t} \wedge \mathbf{m}'_1) = \mathbf{0} \tag{16}$$

where $r = \|\mathbf{m}_1\|/\|\mathbf{m}'_1\|$. Using the fact that $\mathbf{m}'_1$ and $\mathbf{m}'_2$ are orthogonal, 16 can be simplified as:

$$\mathbf{m}'_2 \wedge \mathbf{R}\mathbf{m}_2 + r(\mathbf{m}'_2.\mathbf{t})\mathbf{m}'_1 = \mathbf{0}$$

Finally, if we match a plane $(\mathbf{n}', d')$ in the scene to a plane $(\mathbf{n}, d)$ in the model, the following relationships must hold:

$$\mathbf{n}' = \mathbf{R}\mathbf{n} \tag{17}$$

$$d' = d + \mathbf{t}.\mathbf{R}\mathbf{n} \tag{18}$$

Using equation 17, equation 18 can be rewritten as:

$$d' = d + \mathbf{t}.\mathbf{n}'$$

In practice, due to measurement errors, these relations are not exactly satisfied and we are lead to a mean-square solution. Specifically, given k point matches, we have to solve:

$$\min_{\mathbf{R},\mathbf{t}} \sum_i \|\mathbf{p}'_i - \mathbf{R}\mathbf{p}_i - \mathbf{t}\|^2 \tag{19}$$

Given k line matches:

$$\min_{\mathbf{R},\mathbf{t}} \sum_i \alpha\|\mathbf{v}'_i - \mathbf{R}\mathbf{v}_i\|^2 + (1-\alpha)\|\mathbf{m}'_{2i} \wedge \mathbf{R}\mathbf{m}_{2i} + r_i(\mathbf{t}.\mathbf{m}'_{2i})\mathbf{m}'_{1i}\|^2 \tag{20}$$

where $\mathbf{v}_i = \mathbf{m}_{1i}/\|\mathbf{m}_{1i}\|, \mathbf{v}'_i = \mathbf{m}'_{1i}/\|\mathbf{m}'_{1i}\|$. The choice of α is left to the decision of the user. Finally, given k plane matches, we have to solve:

$$\min_{\mathbf{R}} \sum_i \|\mathbf{n}'_i - \mathbf{R}\mathbf{n}_i\|^2 \tag{21}$$

$$\min_{\mathbf{t}} \sum_i (d'_i - d_i + \mathbf{t}.\mathbf{n}'_i)^2 \tag{22}$$

Closed form solutions for these problems are not easy to obtain because of the constraints on the rotation matrix $\mathbf{R}$. We now make good use of the various representations for rotations presented in Appendix B to come up with either closed form or iterative solutions.

Let us start with the problem of matching planes and consider problem 21. Using the quaternion representation of rotations described in Appendix B, we can rewrite this as:

$$\min_{\mathbf{R}} \sum \mid \mathbf{n}'_i - \mathbf{q} * \mathbf{n}_i * \bar{\mathbf{q}} \mid \tag{23}$$

where vectors and quaternions are identified. Because $\mathbf{q}$ represents a rotation $\mid \mathbf{q} \mid^2 = 1$. Therefore we can multiply 23 by $\mid \mathbf{q} \mid^2$:

$$\min_{\mathbf{R}} \sum (\mid \mathbf{n}'_i - \mathbf{q} * \mathbf{n}_i * \bar{\mathbf{q}} \mid\mid \mathbf{q} \mid)^2$$

and since the quaternion magnitude is compatible with the quaternion product, this is the same as:

$$\min_{\mathbf{R}} \sum \mid \mathbf{n}'_i * \mathbf{q} - \mathbf{q} * \mathbf{n}_i * \bar{\mathbf{q}} * \mathbf{q} \mid^2$$

and since $\bar{\mathbf{q}} * \mathbf{q} = \mid \mathbf{q} \mid^2 = 1$ we finally have:

$$\min_{\mathbf{R}} \sum \mid \mathbf{n}'_i * \mathbf{q} - \mathbf{q} * \mathbf{n}_i \mid^2 \tag{24}$$

It follows from the definition of the quaternion product given in Appendix B that $\mathbf{n}'_i * \mathbf{q} - \mathbf{q} * \mathbf{n}_i$ is a linear function of the 4 coordinates of $\mathbf{q}$. Therefore, there exists a 4×1 vector $\mathbf{a}_i$ such that:

$$\mid \mathbf{n}'_i * \mathbf{q} - \mathbf{q} * \mathbf{n}_i \mid^2 = \mathbf{q}^T \mathbf{a}_i \mathbf{a}_i^T \mathbf{q}$$

where $\mathbf{q}$ denotes here the 4×1 vector attached to the quaternion $\mathbf{q}$. Therefore, equation 24 can be rewritten as:

$$\min_{\mathbf{q}} \mathbf{q}^T \mathbf{A} \quad \mathbf{q} \tag{25}$$

with:

$$\mathbf{A} = \sum \mathbf{a}_i \mathbf{a}_i^T$$

and the constraint on the vector $\mathbf{q}$ is $\|\mathbf{q}\|^2 = 1$. The solution to equation 25 is the eigenvector of unit length of matrix $\mathbf{A}$ corresponding to the smallest eigenvalue. The best translation in equation 22 can then be found using standard least square techniques.

Let us look at the case of points now. Introducing the centroids $\mathbf{p}'$ and $\mathbf{p}$ of the points $\mathbf{p}'_i$ and $\mathbf{p}_i$ and letting $\mathbf{n}'_i = \mathbf{p}'_i - \mathbf{p}'$ and $\mathbf{n}_i = \mathbf{p}_i - \mathbf{p}'$, we rewrite equation 19 as:

$$\min_{\mathbf{R},\mathbf{t}} \sum_i \|\mathbf{n}'_i - \mathbf{R}\mathbf{n}_i + \mathbf{p}' - \mathbf{R}\mathbf{p} - \mathbf{t}\|^2 \tag{26}$$

Expanding each term in 26 and noticing that $\sum \mathbf{n}'_i = \sum \mathbf{n}_i = \mathbf{0}$ we are left with:

$$\min_{\mathbf{R},\mathbf{t}} (k\|\mathbf{p}' - \mathbf{R}\mathbf{p} - \mathbf{t}\|^2 + \sum \|\mathbf{n}'_i - \mathbf{R}\mathbf{n}_i\|^2) \tag{27}$$

The next thing to do is to observe that for any rotation matrix, the first term in the criterion, $\|\mathbf{p}' - \mathbf{R}\mathbf{p} - \mathbf{t}\|^2$ can be made equal to zero by choosing $\mathbf{t} = \mathbf{p}' - \mathbf{R}\mathbf{p}$. Therefore, equation 27 is minimized by finding the best rotation that minimizes the second term, which then determines the translation.The problem for points is thus solved by first computing $\mathbf{R}^*$ minimizing $\sum \|\mathbf{n}'_i - \mathbf{R}\mathbf{r}_i\|^2$, and that can be done using the technique developed for planes, and then setting $\mathbf{t}^* = \mathbf{p}' - \mathbf{R}^*\mathbf{p}$.

We do not have an analytical solution to problem 20 for the lines but we propose two reasonable solutions. The first solution consists in solving for the rotation $\mathbf{R}^*$ by the previous method while using only the first part of the criterion:

$$\sum \|\mathbf{v}'_i - \mathbf{R}\mathbf{v}_i\|^2$$

and then use the second part of the criterion:

$$\sum \|\mathbf{m}'_{2i} \wedge \mathbf{R}^* \mathbf{m}_{2i} + r_i(\mathbf{t}.\mathbf{m}'_{2i})\mathbf{m}'_{1i}\|^2$$

to solve for the translation (this is a standard least-squares problem for $\mathbf{t}$). This corresponds to a choice of $\alpha = 1$ in criterion 20.

Another possibility, which has proven to be extremely fruitful [AF87], is to do a linearized recursive least-square estimation of the rotation, using its exponential representation (see Appendix B), and a recursive least-squares estimation of $\mathbf{t}$. Recursive least-squares estimation techniques arerelated to Kalman filtering techniques [JAZ70,MAY79]. Here we simply want to stress the fact that the exponential representation of rotations allows us very simply to linearize the problem since there are no constraints on the 3×1 vector $\mathbf{r}$ representing a rotation. Therefore we have to compute:

$$d(\mathbf{R}\mathbf{v}_i)/d\mathbf{r} \quad \text{and} \quad d(\mathbf{m}'_{2i} \wedge \mathbf{R}\mathbf{m}_{2i})/d\mathbf{r}$$

which is easily done using the results of Appendix B.

4.4 Controlling the search

There are two basic steps in the search process which is very similar to the one used in Section 2.4 to find Stereo matches. The first is the hypothesis formation, i.e. the search for sets of consistent pairings that provide enough information for the computation of the rigid transformation. The second step is the Prediction of new matches using the estimated transformation and Verification of the correctness of the initial Hypothesis.

4.4.1 Hypothesis formation

This step is crucial because it determines the number of paths that are going to be explored in the interpretation tree before finding an acceptable solution. The main problem is that we need at least two pairings to estimate the transformation or part of it. The number of primitives required is summarized below:

	Translation	Rotation
points	3	3
lines	2	2
planes	3	2

The two lines/planes should not be parallel and the three planes should not meet along a line, otherwise the transformation is underconstrained and cannot be estimated completely.

The hypothesis formation proceeds in three steps:

1. **Selection of a first pairing**: For each primitive in the model, the compatible primitives of the scene are considered. The choice of these primitives cannot make use of the rigidity constraint, only the position invariant features such as the length of the segments, or the area of the surface patches can be used. All these features are highly sensitive to occlusion and therefore should be used carefully and with large tolerances.

2. **Selection of a second pairing**: Given a first pairing (M_1, S_1) and a second model primitive M_2, the candidates for the matching must satisfy the rigidity constraint. For points, this choice is quite simple, the only constraint being that $d(S_1, S_2) = d(M_1, M_2)$.

   For planes, the only constraint is on the angle between the normals, i.e. S_2 must be chosen such that $(\mathbf{n}'_2, \mathbf{n}'_1) = (\mathbf{n}_2, \mathbf{n}_1)$. When two pairs of planes have been matched, the rotation is fixed and the translation is constrained to be parallel to the intersection of the planes M_1 and M_2.

   Let us consider two lines M_1 and M_2 in the model and S_1 and S_2 in the scene represented by their Plücker coordinates $(\mathbf{m}_{11}, \mathbf{m}_{21})$, $(\mathbf{m}_{12}, \mathbf{m}_{22})$, $(\mathbf{m}'_{11}, \mathbf{m}'_{21})$, and $(\mathbf{m}'_{21}, \mathbf{m}'_{22})$. Just as in the case of planes, we have an angle constraint, namely that $(\mathbf{m}_{11}, \mathbf{m}_{12}) = (\mathbf{m}'_{11}, \mathbf{m}'_{12})$, but we also have a metric constraint, for example that their shortest distances $d(M_1, M_2)$ and $d(S_1, S_2)$ be equal.

   These constraints are general in the sense that they do not make any hypothesis about the size of the primitives being matched. In practice, lines are line segments and planes are polygonal faces. In particular, they will not discriminate between two coplanar faces or two aligned segments. Grimson and Lozano-Pérez [GL83]. argue about using constraints obtained from bounds on the components of a vector joining two points on two faces or two line segments in the model. Indeed, if $\mathbf{n}_1$ and $\mathbf{n}_2$ are the two unit normals to the faces and $\mathbf{d}$ the separation vector between two points on each face, then when these two points vary in the faces, the components of $\mathbf{d}$ in the coordinate frame defined by $\mathbf{n}_1$, $\mathbf{n}_2$, and $\mathbf{n}_1 \wedge \mathbf{n}_2$ vary in some intervals. These intervals can be computed once for all when building the model.

When making an hypothesis, these ranges can be used to further filter the list of potential candidates for the second match by choosing a few points on S_1 and S_2, computing the corresponding vectors $\mathbf{d}$, and verifying that their components in the coordinate frame defined by $\mathbf{n}'_1$, $\mathbf{n}'_2$, and $\mathbf{n}'_1 \wedge \mathbf{n}'_2$ fall in the ranges.

This idea can be extended to line segments as well by considering the coordinate frame defined by $\mathbf{m}_{11}$, $\mathbf{m}_{12}$, and $\mathbf{m}_{11} \wedge \mathbf{m}_{12}$. Our experience has shown that in general, these constraints do not add very much to the power of the original rigidity constraint.

3. **Estimation of the transformation**: The transformation is estimated (or partially estimated in the case of planar patches) using the techniques described in the previous Section. We mentioned the fact that some primitives may not have a canonic orientation, therefore the transformation estimated from an initial hypothesis is not unique and several equivalent transformations are generated (two for lines and planes).

   Another important item is the order in which the model primitives are considered for matching. Non-interesting branches of the interpretation tree might be explored if the order is not carefully determined. Consider for example the case in which the first two primitives are parallel planes. The estimated rotation is arbitrary and the rotation error vanishes. A large subtree may be explored based on a wrong estimation of the transformation. A number of rules can be applied in the ordering of primitives:

   - Small primitives (in terms of length or area) should be avoided because of their probable sensitivity to measurement noise
   - The first two or three primitives must be linearly independent in order to avoid indetermination
   - If local symmetries exist in the object, the primitives that best discriminate between almost equivalent positions of the object should be among the first ones considered for matching. Notice that in some cases this might contradict the first two rules

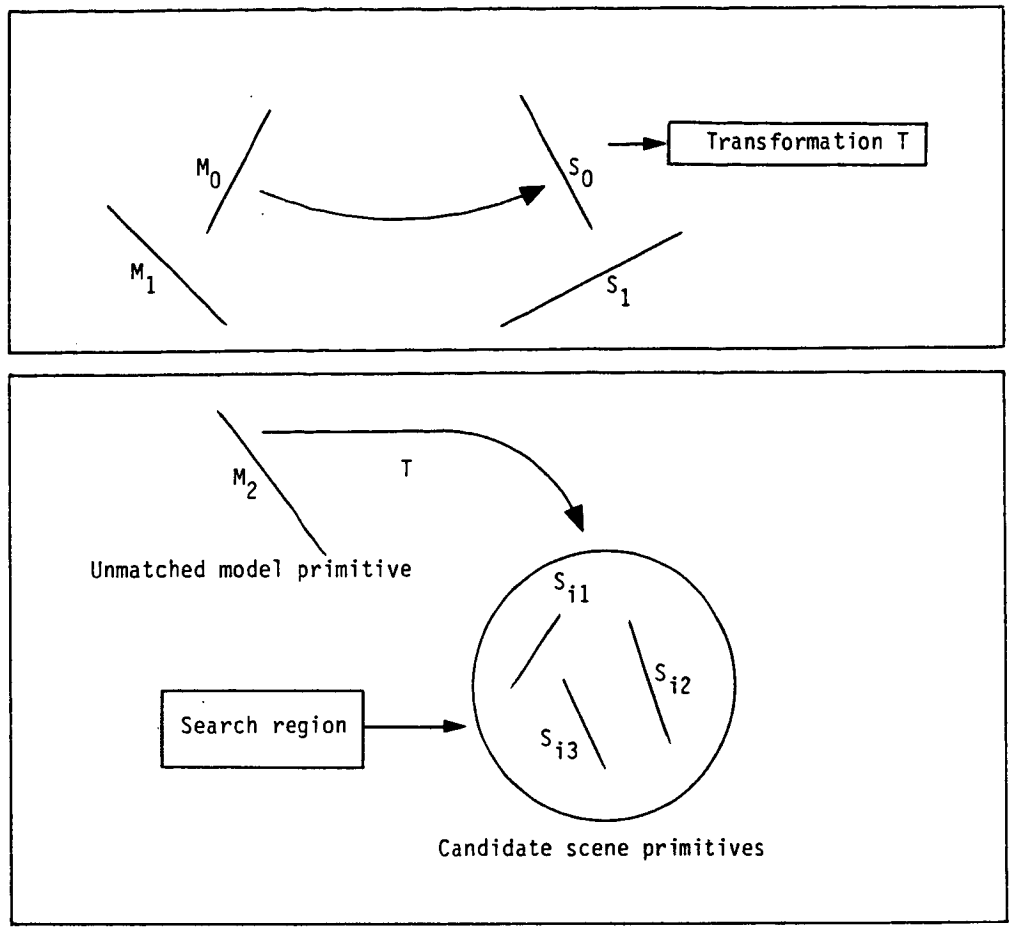


Figure 51: Finding potential matches

4.4.2 Prediction and verification

In this step, given an initial hypothesis and the associated transformation $\mathbf{T}$, we want to *predict* the set of candidate scene primitives that can be matched with model primitives in order to *verify* the validity of the initial hypothesis.

The basic way of doing this is by applying the transformation to every unmatched model primitive M and find the scene primitives which are close enough to $\mathbf{T}(M)$ (see Figure 51). An important issue here is that we want to avoid a sequential exploration of the scene description for each model primitive because the rigidity constraint is used precisely to reduce the number of potential candidates for a match.

One solution to this problem, which we used several times in this paper, is the grouping of scene primitives into buckets corresponding to discretized values of the parameters defining the transformation. More precisely, the scene is divided into a number of cubic buckets. To each scene primitive is attached the list of buckets it intersects, and to each

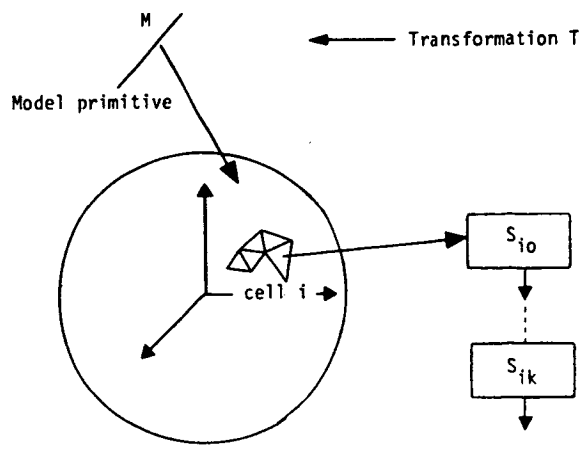


Figure 52: Orientation buckets

bucket is attached the list of scene primitives that intersect it. Moreover, the primives can also be grouped by orientation buckets as follows. The gaussian sphere is used to representation orientations of lines or planes and can be tesselated in a number of ways. Each tessel is a bucket and all scene primitives whose orientations fall into that bucket are attached to it (Figure 52).

Returning to our unmatched model primitive M, we compute $\mathbf{T}(M)$, from its orientation determine which bucket of the gaussian sphere it falls into. This in turn determines which scene primitives are candidates for the match. If the translation is known, the corresponding buckets in the scene determine another set of candidates for the match. The final candidates are the intersection of these two lists. This basically controls the breadth of the exploration of the interpretation tree and allows to find all possible positions of the model in the scene satisfying the constraint that the error is less than some threshold and the number of model faces assigned to NIL below another threshold.

This second threshold can be used to also control the depth of the exploration by noticing that if at some point in the exploration we reach a point where, even if all the remaining model primitives are matched to scene primitives, the threshold cannot be reached, then the exploration might as well stop.

Results obtained by a system that uses some of these ideas are shown in Figures 53- 55 which are taken from [FH86]. For an early implementation of some of these ideas in the planar case see [AF86].

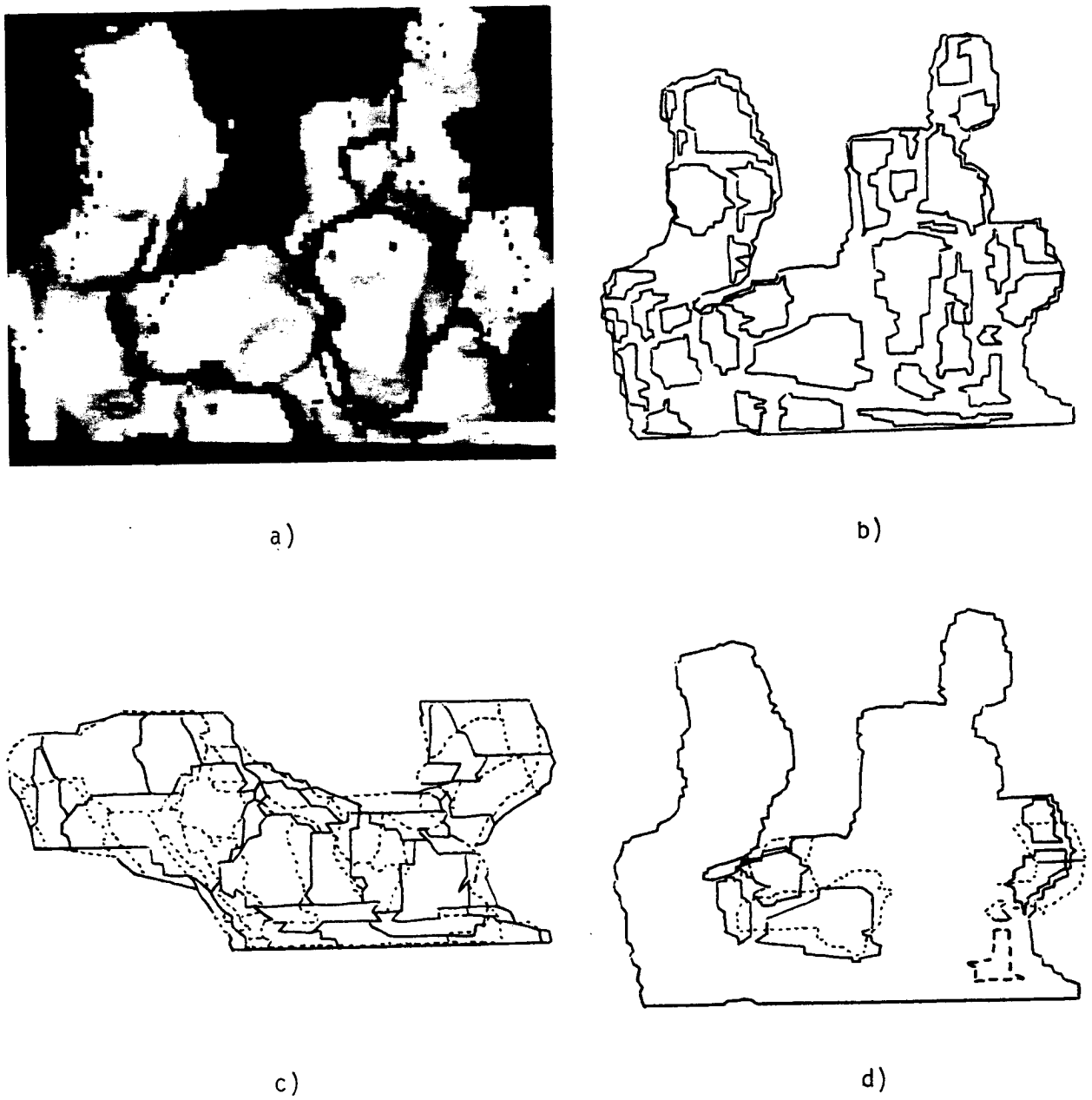


Figure 53: Results of the recognition and locating algorithm: a) Image of the normals to the scene. b) Scene segmentation in planar patches. c) First identified model after rotation with the estimated rotation matrix. d) Superimposition of identified scene and model primitives.

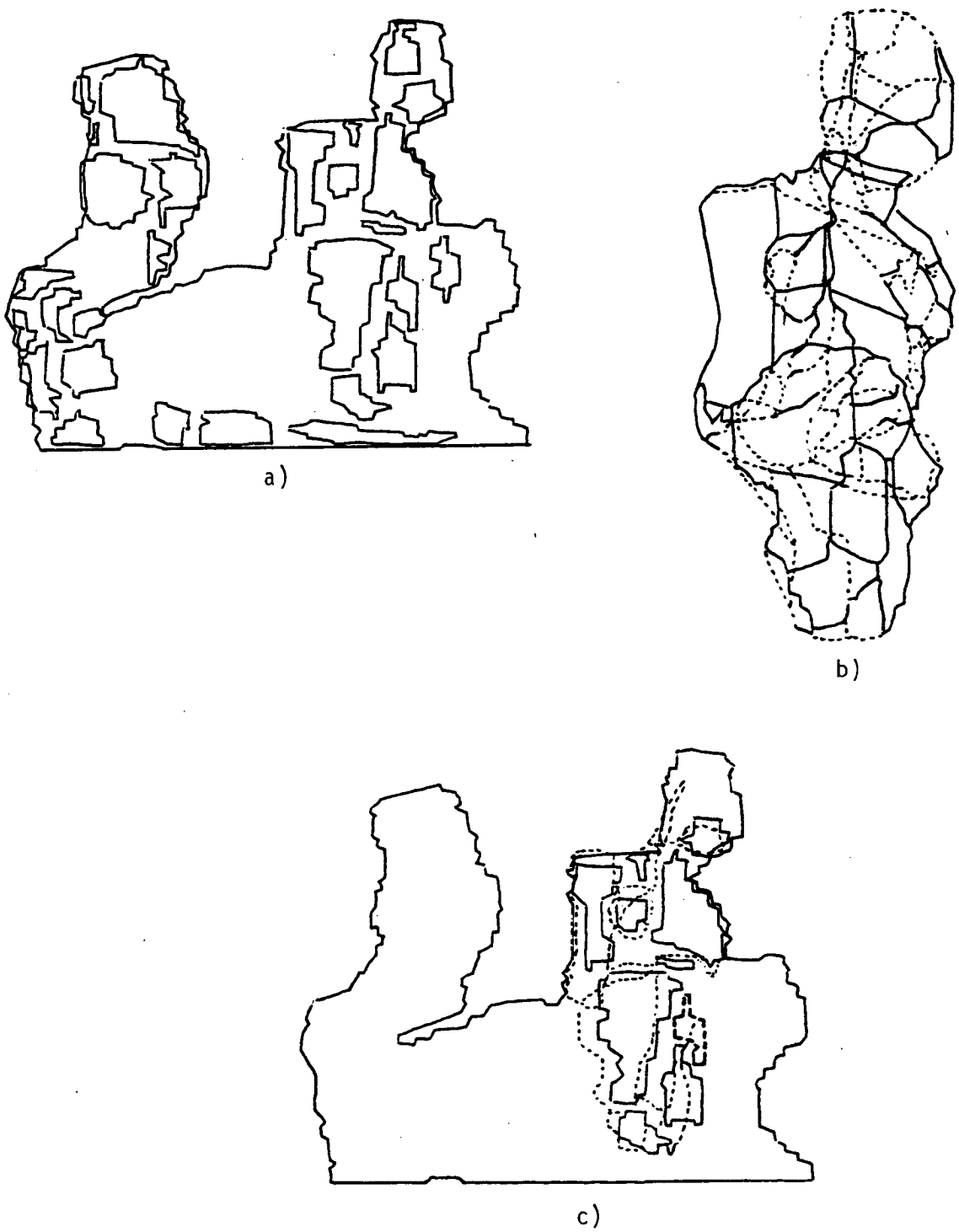


Figure 54: Results of the recognition and locating algorithm (cont.): a) Scene segmentation in planar patches. b) Second identified model after rotation with the estimated rotation matrix. c) Superimposition of identified scene and model primitives.

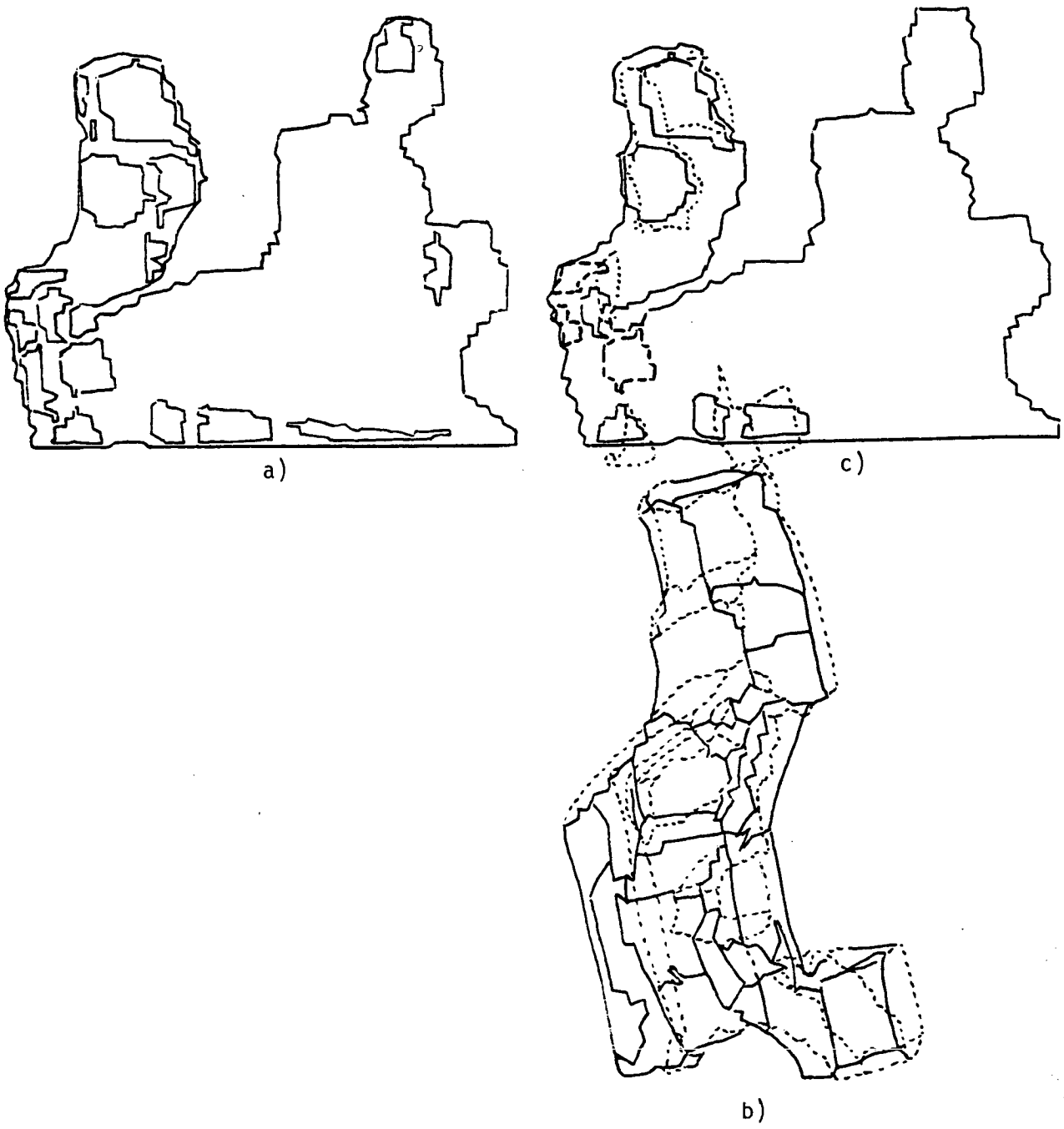


Figure 55: Results of the recognition and locating algorithm (cont.): a) Scene segmentation in planar patches. b) Third identified model after rotation with the estimated rotation matrix. c) Superimposition of identified scene and model primitives.

4.5 What needs to be worked on

There are an enormous number of things that need to be worked on in this area, some of them we referred to briefly in Section 3.4. Problems are of course difficult to isolate from the problems in shape representation but we can identify a number of "hot" areas which need special attention:

1. the need to take into account explicitely in the recognition and localization process the uncertainty on the primitives which are matched. Some work has begun to appear [LC85,CRO86,DUR86,FAF86,SC86] but a lot remains to be done. In particular, nothing has yet been proposed to use results in probabilistic geometry where we believe a lot of potentially useful results can be found.

2. with respect to geometry, we believe that the community badly needs systems capable of manipulating symbolic geometric information. Again, some work is going on in this area [CHO84,MUN86,WU78], but more effort needs to be put into it. We need a geometric equivalent to the MACSYMA system.

3. it is also important to demonstrate systems which have the capability to recognize and locate rigid objects from data coming from stereo and structure from motion where noise, matching errors and sparseness of 3D data make the problem a lot harder than in the case of dense range data. We believe that this is a somewhat easy problem but which should be clearly solved in order to make passive 3D Vision a reality.

4. in parallel, more complicated problems such as articulated rigid objects should also be investigated [GRI87]. These problems have obvious practical applications and are probably a good way of attacking an even more difficult problem which we also believe time has come to tackle with some hopes of success:

5. the problem of recognizing classes of objects as opposed to specific instances of one object. Previously, we gave the example of the class of chairs, there are obviously many other examples. This problem raises a number of fascinating questions such as the relation of shape and function and the relation of the Language and our Perception of the 3D world. We believe that Vision theory and practice have now reached a stage where these questions can be asked without coming up, like so often in the past, with solipsistic systems.

Acknowledgments : I want to thank a number of people without whom this paper could not have been written: Nicholas Ayache, Jean Daniel Boissonnat, Michael Brady, Rodney Brooks, Eric Grimson, Martial Hebert, Ellen Hildreth, Thomas Huang, Elizabeth Le Bras, Francis Lustman, and Giorgio Toscani.

I also want to thank Nathalie Rocher for her superb job at preparing and editing the manuscript and Figures.

Last but not least, this work was partially supported by ESPRIT Project P940.

A Computing the epipolar geometry

We describe briefly the epipolar geometry of a stereo pair. Let us make a simple geometric remark, helping ourselves with Figure 3. Indeed, in this figure we can see that given m_1 in retina plane 1, all possible physical points M which may have produced m_1 are on the infinite half-line C_1m_1. As a direct consequence, all possible correspondents m_2 of m_1 in plane 2 are located on the image, through the second imaging system, of this infinite half-line. This image is also an infinite half-line ($ep2$) starting at point E_2, intersection of the line going through C_1 and C_2 and the plane 2. E_2 is called the Epipole of plane 2 with respect to plane 1, and the line ($ep2$) is called the epipolar line of point m_1 in the plane 2. The corresponding constraint is that for a given point m_1 in plane 1, its possible correspondents in plane 2 all lie on a half line in plane 2 (see Figure 13).

Therefore we have reduced the dimension of our search space from 2 to 1 dimension. The epipolar constraint is of course symmetric and for a given point m_2 in plane 2, its possible correspondents in plane 1 all lie on a half line ($ep1$) starting at the epipole E_1, intersection of the line C_1C_2 with plane 1. ($ep1$) and ($ep2$) are the intersections of the plane C_1MC_2, called the epipolar plane, with planes 1 and 2, respectively.

Of course, when plane 1 or plane 2, or both, are parallel to this line, one (or both) epipoles go to infinity and epipolar lines in one plane (or both) become parallel. The situation where both planes are parallel to the line C_1C_2 is often assumed because of its simplicity. But, in practice, it is difficult to align precisely and in a stable way the two optical systems and we show next that the problem of computing the epipolar lines is just as simple in the general case.

The coordinates of C_i ($i = 1,\ 2$), the two optical centers in the world reference frame are obtained by solving the following systems of linear equations :

$$\mathbf{M}_i\mathbf{C}_i = \mathbf{0} \qquad i = 1, 2$$

where $\mathbf{C}_i = [x_i, y_i, z_i, 1]^t$ is the coordinate vector of $C_i, i = 1, 2$. Since each epipole E_i is the image by the ith camera of the other camera's optical center C_j ($j \neq i$), the image coordinates of the epipoles E_i are obtained by applying matrices $\mathbf{M}_i$ to the vectors $\mathbf{C}_j (i, j = 1,\ 2,\ i \neq j)$.

Let us now show how, for a given point m_1 in plane 1, its corresponding epipolar line can be easily computed from its coordinates. Using again equations 1 for camera 1, slightly modified to deal only with 3×1 column vectors, we have:

$$\begin{aligned} \mathbf{l}_1^t\mathbf{X} - u_1\mathbf{l}_3^t\mathbf{X} + l_{14} - u_1 l_{34} &= 0 \\ \mathbf{l}_2^t\mathbf{X} - v_1\mathbf{l}_3^t\mathbf{X} + l_{24} - v_1 l_{34} &= 0 \end{aligned}$$

These are the equations of two planes whose intersection is the line C_1m_1. A vector $\mathbf{n}$ parallel to the line is given by the cross product of the two normal vectors $\mathbf{l}_1 - u_1\mathbf{l}_3$ and $\mathbf{l}_2 - v_1\mathbf{l}_3$:

$$\mathbf{n} = u_1\mathbf{l}_2 \wedge \mathbf{l}_3 + v_1\mathbf{l}_3 \wedge \mathbf{l}_1 + \mathbf{l}_1 \wedge \mathbf{l}_2 = u_1\mathbf{g} + v_1\mathbf{h} + \mathbf{k}$$

The line C_1m_1 is described by the vectors $\mathbf{c}_1 + \lambda\mathbf{n}$ where λ varies from $-\infty$ to $+\infty$. The epipolar line of m_1 is the image of that line through camera 2, therefore, from equation 1, the projective coordinates $\mathbf{u}_2 = [U_2, V_2, S_2]^t$ of a point on that epipolar line are given by :

$$\mathbf{u}_2 = \mathbf{M}_2 \begin{bmatrix} \mathbf{c}_1 \\ 1 \end{bmatrix} + \lambda_{\mathbf{n}} = \mathbf{e}_2 + \lambda\mathbf{M}_2'\mathbf{n}$$

where $\mathbf{M}_2'$ is the (3×3) matrix obtained by dropping the last column of $\mathbf{M}_2$. Letting $\mathbf{a} = [a_1, a_2, a_3]^t = \mathbf{M}_2'\mathbf{n}$, we see that both the epipole E_2 represented by $\mathbf{e}_2$ and the point represented by a belong to the epipolar line.

Letting $\mathbf{G} = \mathbf{M}_2'\mathbf{g}, \mathbf{H} = \mathbf{M}_2'\mathbf{h}$, and $\mathbf{K} = \mathbf{M}_2'\mathbf{k}$, we have :

$$\mathbf{a} = u_1\mathbf{G} + v_1\mathbf{H} + \mathbf{K}$$

and therefore :

$$\mathbf{a} = \mathbf{F} \quad \mathbf{u}$$

where $\mathbf{F}$ is the 3×3 matrix $[\mathbf{G}, \mathbf{H}, \mathbf{K}]$ and $\mathbf{u} = [u_1, v_1, 1]^t$.

The projective equation of the epipolar line attached to the point m_1 is given by the determinant:

$$\begin{bmatrix} U & u_{2e} & a_1 \\ V & V_{2e} & a_2 \\ S & S_{2e} & a_3 \end{bmatrix} = 0$$

Therefore, the equation of the line can be written as :

$$(V_{2e}a_3 - S_{2e}a_2)U + (S_{2e}a_1 - U_{2e}a_3)V + (U_{2e}a_2 - V_{2e}a_3)S = 0$$

The cartesian equation is of course :

$$(V_{2e}a_3 - S_{2e}a_2)u + (S_{2e}a_1 - U_{2e}a_3)v + (U_{2e}a_2 - V_{2e}a_3) = 0$$

Notice that this encompasses the case where the epipole E_2 is at infinity ($S_{2e} = 0$) or at a finite distance ($S_{2e} \neq 0$). The coefficients of the equation of the line are affine functions of the coordinates u_1, v_1 of point m_1, with coefficients depending only on matrixes $\mathbf{M}_1$ and $\mathbf{M}_2$, and are therefore easily computed.

B Representing 3D Rotations

B.1 Orthogonal matrixes

An orthogonal matrix is a real square matrix satisfying:

$$\mathbf{R}\mathbf{R}^T = \mathbf{I} \tag{28}$$

This implies that $det(\mathbf{R})^2 = 1$ and therefore that $det(\mathbf{R}) = \pm 1$. It can be proved that every 3×3 orthogonal matrix of determinant equal to 1 is the matrix of a rotation and inversely that every rotation can be represented as such a matrix.

Equation 28 implies that some very strict constraints must be satisfied by the row and column vectors of $\mathbf{R}$. More precisely, it implies that the row (column) vectors are mutually orthogonal and of unit lengths, i.e if we denote by $\mathbf{r}_i (i = 1,2,3)$ the column vectors, they satisfy:

$$\mathbf{r}_i^T \mathbf{r}_j = \delta_{ij} \qquad i,j = 1,2,3, i \leq j \tag{29}$$

where δ_{ij} is the usual Kronecker symbol:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

The relations 29 can be considered as a set of 6 independent quadratic constraints that are satisfied by the elements of an orthogonal matrix $\mathbf{R}$.

B.2 Rodrigues formula

Given a rotation of angle θ with respect to an axis $\mathbf{u}$ (a vector of unit length), then there is a simple relationship between the orthogonal matrix $\mathbf{R}$ representing the rotation and θ and $\mathbf{u}$. This relationship is known as the Rodrigues formula from the 19th century french mathematician who published it first [ROD40]. If we denote by $\mathbf{U}$ the matrix associated with $\mathbf{u} = [u_1, u_2, u_3]^T$:

$$\begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$$

then it can be easily verified that for any vector $\mathbf{x}$ we have, $\mathbf{u} \wedge \mathbf{x} = \mathbf{U}\mathbf{x}$. Rodrigues formula then says that:

$$\mathbf{R} = \mathbf{I} + sin(\theta)\mathbf{U} + (1 - cos(\theta))^2\mathbf{U}^2 \tag{30}$$

B.3 Quaternions

Quaternions have been found useful in Robotics and Vision [PW83,FH86,HOR86]. They can be understood very simply by looking at them as 4×1 real numbers that form a vector space in the usual sense and on which we define a multiplication to turn that vector space into a noncommutative field.

We therefore consider a quaternion $\mathbf{q}$ as being either a 4×1 vector $[q_1, q_2, q_3, q_4]^T$, or as a pair $(s, \mathbf{v})$ where s is a real number equal to q_1, and $\mathbf{v}$ is the vector $[q_2, q_3, q_4]^T$. In this notation, a quaternion is very similar to a complex number, s being like the real part, and $\mathbf{v}$ like the imaginary part. A real number x is readily identified with the quaternion $(x, \mathbf{0})$, the product of two real numbers is the real part of the the product of the corresponding quaternions:

$$(x, \mathbf{0}) * (x', \mathbf{0}) = (xx', \mathbf{0})$$

and a 3×1 real vector $\mathbf{v}$ is readily identified with the quaternion $(0, v)$.

We now define the product $*$ of two quaternions $\mathbf{q}$ and $\mathbf{q}'$ as follows:

$$\mathbf{q} * \mathbf{q}' = (ss' - \mathbf{v}.\mathbf{v}', s\mathbf{v}' + s'\mathbf{v} + \mathbf{v} \wedge \mathbf{v}') \tag{31}$$

The definitions of the conjugate and the magnitude of a quaternion are very similar to the ones for the complex numbers:

$$\overline{\mathbf{q}} = (s, -\mathbf{v}) \quad \text{and} \quad |\, \overline{\mathbf{q}} \,|^2 = \overline{\mathbf{q}} * \mathbf{q} = \mathbf{q} * \overline{\mathbf{q}} = (s^2 + \|\mathbf{v}\|^2, \mathbf{0}) = (\|\mathbf{q}\|^2, \mathbf{0})$$

In this formula, we have used $|$ for the quaternion magnitude and $\|$ for the usual euclidean norm. It can be easily verified that the magnitude is compatible with the product in the sense that:

$$|\, \mathbf{q} * \mathbf{q}' \,| = |\, \mathbf{q} \,|\, |\, \mathbf{q}' \,|$$

This is just about all we need to know about quaternion in order to use them to represent 3D rotations. Indeed, a rotation of angle θ with respect to an axis $\mathbf{u}$ (a vector of length 1), can be represented by the two quaternions $\mathbf{q} = (s, \mathbf{v})$ and $-\mathbf{q}$, where:

$$s = cos(\theta/2) \tag{32}$$

$$\mathbf{v} = sin(\theta/2)\mathbf{u} \tag{33}$$

Notice that $\|\mathbf{q}\| = 1$.

It should not be surprising that there are two quaternions for one rotation, since a rotation of angle θ with respect to an axis $\mathbf{u}$ is the same as a rotation of angle $2\pi - \theta$ with respect to the axis $-\mathbf{u}$. Looking at the previous formula, the two rotations correspond precisely to the two quaternions $\mathbf{q}$ and $-\mathbf{q}$. Inversely, for a quaternion $\mathbf{q}$ of magnitude 1, it is clear that there exists a unique rotation defined by the two formulas 33.

The correspondence between rotations and quaternions of unit magnitude is even deeper, since it preserves the group operation. Indeed, given two rotations 1 and 2 and the associated quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ (for each rotation choose any one of the two possible quaternions, it does not matter which one you choose), then the product of rotation 1 with rotation 2 (that is the rotation obtained by first applying 2 and then 1), corresponds to the products $\mathbf{q}_1 * \mathbf{q}_2$ and $-\mathbf{q}_1 * \mathbf{q}_2$.

Since the product of rotations is <u>not</u> commutative unless they have the same axis, then this shows (as also does formula 31) that the product of two quaternions is not commutative.

This correspondence between rotations and quaternions allows us to derive a very useful formula. Let $\mathbf{R}$ be the orthogonal matrix representing a rotation of angle θ with respect to an axis $\mathbf{u}$ (a vector of length 1), $\mathbf{q}$ one of the two corresponding quaternions, and $\mathbf{w}$ a 3×1 vector. Then we can write:

$$\mathbf{Rw} = \mathbf{q} * \mathbf{w} * \overline{\mathbf{q}} \tag{34}$$

In this formula, we have identified 3×1 vectors and the corresponding quaternion, i.e, the formula should be read:

$$(0, \mathbf{Rw}) = \mathbf{q} * (0, \mathbf{w}) * \overline{\mathbf{q}}$$

Formula 34 allows us to derive the relationship between the coefficients of the rotation matrix $\mathbf{R}$ and the coordinates of the quaternions $\mathbf{q}$ and $-\mathbf{q}$ representing it. If we write $\mathbf{q} = [s, l, m, n]^T$ where $\mathbf{v} = [l, m, n]^T$ it is easy to derive, using formula 31 that:

$$\mathbf{q} * \mathbf{w} * \overline{\mathbf{q}} = (0,\ (\mathbf{v}.\mathbf{w})\mathbf{v} + (s^2 - \|\mathbf{v}\|^2)\mathbf{w} + 2s\ \mathbf{v} \wedge \mathbf{w}) \tag{35}$$

We can also write:

$$\mathbf{v} \wedge \mathbf{w} = \tilde{\mathbf{v}}\mathbf{w}$$

with:

$$\tilde{\mathbf{v}} = \begin{bmatrix} 0 & -n & m \\ n & 0 & -l \\ -m & l & 0 \end{bmatrix}$$

and:

$$(\mathbf{v}.\mathbf{w})\mathbf{v} = \mathbf{A}\mathbf{w}$$

where $\mathbf{A}$ is a 3×3 matrix given by:

$$A = [l\mathbf{v}, m\mathbf{v}, n\mathbf{v}] = \begin{bmatrix} l^2 & lm & ln \\ lm & m^2 & mn \\ ln & mn & n^2 \end{bmatrix}$$

Writing that the imaginary part of 35 is equal to $\mathbf{R}\mathbf{w}$, we find that:

$$R = \begin{bmatrix} s^2 + l^2 - m^2 - n^2 & 2(lm - sn) & 2(ln + sm) \\ 2(mn + sn) & s^2 - l^2 + m^2 - n^2 & 2(mn - sl) \\ 2(ln - sn) & 2(mn + sl) & s^2 - l^2 - m^2 - n^2 \end{bmatrix} \tag{36}$$

From 36, it appears that the coefficients of $\mathbf{R}$ are polynomial functions of the coordinates of $\mathbf{q}$.

B.4 Antisymmetric matrices

For every orthogonal matrix $\mathbf{R}$, these exists a unique antisymmetric matrix $\mathbf{H}$ such that:

$$\mathbf{R} = e^{\mathbf{H}}$$

where matrix exponentials are defined as usual. Matrix $\mathbf{H}$ can therefore be written as:

$$\mathbf{H} = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}$$

The three dimensional vector $\mathbf{r} = [a, b, c]^T$ has some useful properties. Its direction is that of the axis of rotation and its squared norm is equal to the rotation angle squared. Moreover, matrix $\mathbf{H}$ represents the cross-product with vector $\mathbf{r}$, which we denote by $c(\mathbf{r})$. By this we mean:

$$\mathbf{Hx} = c(\mathbf{r})\mathbf{x} = \mathbf{r} \wedge \mathbf{x}$$

A simple justification of the fact that every matrix $e^{\mathbf{H}}$, with $\mathbf{H}$ an antisymmetric matrix, is orthogonal is the following:

$$(e^{\mathbf{H}})^T = e^{\mathbf{H}^T} = e^{-\mathbf{H}} = (e^{\mathbf{H}})^{-1}$$

We also know [10] that the eigenvalues a and b of $\mathbf{R}$ and $\mathbf{H}$ are related by:

$$a = e^b$$

Since it is well known that the eigenvalues of $\mathbf{R}$ are 1, $e^{-i\theta}$ and $e^{-i\theta}$, where θ is the rotation angle. The eigenvalues of matrix $\mathbf{H}$ are easily found to be equal to 0, $i(a_2 + b_2 + b_2)^{1/2}$ and $-i(a^2 + b^2 + b^2)^{1/2}$. Therefore we have:

$$\theta = \pm(a^2 + b^2 + b^2)^{1/2}$$

Let us now consider an eigenvector of matrix $\mathbf{H}$ associated with eigenvalue 0. Since matrix $\mathbf{H}$ represents the vector product with vector $\mathbf{v} = [a, b, c]^T$, $\mathbf{v}$ is such a vector:

$$\mathbf{Hv} = \mathbf{0}$$

$\mathbf{v}$ is also an eigenvector of matrix $\mathbf{R}$ associated with eigenvalue 1, as can be easily verified by using the formula:

$$\mathbf{R} = e^{\mathbf{H}} = \mathbf{I} + \mathbf{H}/1! + \mathbf{H}^2/2! + ...$$

Therefore:

$$\mathbf{Rv} = \mathbf{v}$$

and $\mathbf{v}$ gives the direction of the axis of rotation. $\mathbf{H}$ is thus a very compact representation of the rotation in terms of its axis and angle. Another property of this representation can be deduced from a theorem in [GAN77] which states that if we compute the Lagrange-Sylvester polynomial p of the exponential function for the eigenvalues of $\mathbf{H}$, i.e the polynomial such that:

$$\begin{aligned} p(0) &= e^0 = 1 \\ p(i\theta) &= e^{i\theta} \\ p(-i\theta) &= e^{-i\theta} \end{aligned}$$

then we have the nice relationship:

$$e^{\mathbf{H}} = p(\mathbf{H})$$

it can be easily verified that:

$$p(\mathbf{H}) = \mathbf{I} + (sin\theta/\theta)\mathbf{H} + ((1 - cos\theta)/\theta^2)\mathbf{H}^2$$

which is precisely the well-known Rodrigues formula [ROD40].

B.5 Deriving functions of a rotation matrix

B.5.1 Key remark

As we saw in Section B.1, a rotation matrix is orthogonal:

$$\mathbf{R}\mathbf{R}^T = \mathbf{I} \tag{37}$$

Let $(x_1, x_2, x_3) \rightarrow \mathbf{R}$ be a parametrization of class C^1. Deriving 37 with respect to x_i, we obtain:

$$\frac{\partial \mathbf{R}}{\partial x_i}\mathbf{R}^T + \mathbf{R}\frac{\partial \mathbf{R}^T}{\partial x_i} = 0 \tag{38}$$

Equation 38 shows that matrix $\mathbf{B}_i = \frac{\partial \mathbf{R}}{\partial x_i}\mathbf{R}^T$ is antisymmetric:

$$\mathbf{B}_i + \mathbf{B}_i^T = \mathbf{0}$$

Since clearly $\frac{\partial \mathbf{R}}{\partial x_i} = \mathbf{B}_i\mathbf{R}$, we can always write:

$$\frac{\partial \mathbf{ROM}}{\partial x_i} = \mathbf{b}_i \wedge \mathbf{ROM}$$

Computing $\frac{d\mathbf{ROM}}{d\mathbf{x}}$ where $\mathbf{x} = [x_1, x_2, x_3]^T$:

$$\frac{d\mathbf{ROM}}{d\mathbf{x}} = [\frac{\partial \mathbf{ROM}}{\partial x_1} \frac{\partial \mathbf{ROM}}{\partial x_2} \frac{\partial \mathbf{ROM}}{\partial x_3}]$$
$$= [\mathbf{b}_1 \wedge \mathbf{ROM} \quad \mathbf{b}_2 \wedge \mathbf{ROM} \quad \mathbf{b}_3 \wedge \mathbf{ROM}]$$

Therefore:

$$\frac{d\mathbf{ROM}}{d\mathbf{x}}\mathbf{v} = (v_1\mathbf{b}_1 + v_2\mathbf{b}_2 + v_3\mathbf{b}_3) \wedge \mathbf{ROM} \tag{39}$$

where $\mathbf{v} = [v_1, v_2, v_3]^T$.

Now, the problem is to compute the matrixes $\mathbf{B}_i$ for the previous exponential representation.

B.5.2 Exponential representation

We use the representation $\mathbf{R} = e^{\mathbf{H}}$ where $\mathbf{H}$ is antisymmetric.

$$\mathbf{H} = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}$$

The vector $\mathbf{r} = [a, b, c]^T$ is parallel to the axis of rotation and $\|r\|^2 = \theta^2$, θ is the angle of rotation. Let us derive a number of properties of matrix $\mathbf{H}$ which will be used in what follows. If $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ are the three unit vectors defining the standard coordinate system, we will also use the antisymmetric matrixes $\tilde{\mathbf{e}}_1$, $\tilde{\mathbf{e}}_2$, $\tilde{\mathbf{e}}_3$. For example, we have:

$$\tilde{\mathbf{e}}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

It is easy to show that:

$$\frac{\partial \theta}{\partial a} = \frac{a}{\theta} \quad \text{and} \quad \frac{\partial \mathbf{H}}{\partial a} = \tilde{\mathbf{e}}_1$$

Deriving with respect to b, and c yields similar relationships.

Next, we have:

$$\mathbf{H}^2 = \begin{bmatrix} -(b^2 + c^2) & ab & ac \\ ab & -(c^2 + a^2) & bc \\ ac & bc & -(a^2 + b^2) \end{bmatrix}$$

From this we can deduce that:

$$\mathbf{H}^3 = -\theta\mathbf{H} \tag{40}$$

and:

$$\mathbf{H}^2\tilde{\mathbf{e}}_1\mathbf{H} = -a\mathbf{H}^2 \tag{41}$$

Notice that $\mathbf{H}^2\tilde{\mathbf{e}}_1\mathbf{H}$ is an even matrix.

Similarly, we have:

$$\mathbf{H}\tilde{\mathbf{e}}_1 = \begin{bmatrix} 0 & b & c \\ 0 & -a & 0 \\ 0 & 0 & -a \end{bmatrix}$$

and therefore:

$$\mathbf{H}\tilde{\mathbf{e}}_1\mathbf{H} = \begin{bmatrix} 0 & ac & -ab \\ -ac & 0 & a^2 \\ ab & -a^2 & 0 \end{bmatrix} = -a\mathbf{H}$$

Let us rewrite Rodrigues formula (equation 30):

$$\mathbf{R} = \mathbf{I} + f(\theta)\mathbf{H} + g(\theta)\mathbf{H}^2 \tag{42}$$

where:

$$f(\theta) = \frac{sin\theta}{\theta} \qquad g(\theta) = \frac{1 - cos\theta}{\theta^2}$$

Verifying that $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ on this formula yields a useful relationship between f and g:

$$\mathbf{R}^T = \mathbf{I} - f(\theta)\mathbf{H} + g(\theta)\mathbf{H}^2$$

Therefore (we use equation 40):

$$\mathbf{R}\mathbf{R}^T = \mathbf{I} + (2g(\theta) - f^2(\theta) - \theta^2 g^2(\theta))\mathbf{H}^2$$

Therefore, this yields:

$$2g(\theta) - f^2(\theta) - \theta^2 g^2(\theta) = 0 \tag{43}$$

Let us now compute $\mathbf{R}\frac{\partial \mathbf{R}^T}{\partial a}$, using equation 42):

$$\frac{\partial \mathbf{R}}{\partial a} = a\frac{f'(\theta)}{\theta}\mathbf{H} + a\frac{g'(\theta)}{\theta}\mathbf{H}^2 + f(\theta)\tilde{\mathbf{e}}_1 + g(\theta)(\mathbf{H}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_1\mathbf{H})$$

We notice that $\mathbf{H}^2$ and $\mathbf{H}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_1\mathbf{H}$ are symmetric matrices, and write:

$$\frac{\partial \mathbf{R}^T}{\partial a} = -a\frac{f'(\theta)}{\theta}\mathbf{H} - f(\theta)\tilde{\mathbf{e}}_1 + a\frac{g'(\theta)}{\theta}\mathbf{H}^2 + g(\theta)(\mathbf{H}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_1\mathbf{H})$$

from this, we compute:

$$\begin{aligned}\mathbf{R}\frac{\partial \mathbf{R}^T}{\partial a} = \; & -\theta a\left(\frac{f'(\theta)}{\theta a} + g'(\theta)f(\theta) - g(\theta)f'(\theta)\right)\mathbf{H} \\ & -f(\theta)\tilde{\mathbf{e}}_1 + f(\theta)g(\theta)\mathbf{H}\tilde{\mathbf{e}}_1\mathbf{H} \\ & +\frac{a}{\theta}(g'(\theta) - f(\theta)f'(\theta) - \theta^2 g'(\theta)g(\theta))\mathbf{H}^2 \\ & +g(\theta)(\mathbf{H}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_1\mathbf{H}) \\ & -f^2(\theta)\mathbf{H}\tilde{\mathbf{e}}_1 + g^2(\theta)\mathbf{H}^2(\mathbf{H}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_1\mathbf{H}) \end{aligned} \tag{44}$$

Using relation 41, we can rewrite the previous equation as:

$$\begin{aligned}\mathbf{R}\frac{\partial \mathbf{R}^T}{\partial a} = \; & -\theta a\left(\frac{f'(\theta)}{\theta^2} + g'(\theta)f(\theta) - g(\theta)f'(\theta)\right)\mathbf{H} - f(\theta)\tilde{\mathbf{e}}_1 + f(\theta)g(\theta)\mathbf{H}\tilde{\mathbf{e}}_1\mathbf{H} \\ & +\frac{a}{\theta}(g'(\theta) - f(\theta)f'(\theta) - \theta^2 g'(\theta)g(\theta) - \theta g^2(\theta))\mathbf{H}^2 \\ & +g(\theta)(\mathbf{H}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_1\mathbf{H}) - (f^2(\theta) + \theta^2 g^2(\theta))\mathbf{H}\tilde{\mathbf{e}}_1 \end{aligned} \tag{45}$$

Rewriting $\mathbf{H}\tilde{\mathbf{e}}_1$ as the sum of an odd and even component:

$$\mathbf{H}\tilde{\mathbf{e}}_1 = \frac{\mathbf{H}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_1\mathbf{H}}{2} + \frac{\mathbf{H}\tilde{\mathbf{e}}_1 - \tilde{\mathbf{e}}_1\mathbf{H}}{2}$$

the coefficient of $\mathbf{H}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_1\mathbf{H}$ in equation 45 is:

$$g(\theta) - \frac{f^2(\theta)}{2} - \frac{\theta^2 g^2(\theta)}{2}$$

which is equal to zero, thanks to equation 43. If we derive 43 with respect to θ, we obtain:

$$g'(\theta) - f(\theta)f'(\theta) - \theta^2 g(\theta)g'(\theta) - \theta g^2(\theta) = 0$$

which shows that the coefficient of $\mathbf{H}^2$ in 45 is zero.

We are therefore left with the sum of four odd terms, in $\mathbf{H}$, $\tilde{\mathbf{e}}_1$, $\mathbf{H}\tilde{\mathbf{e}}_1\mathbf{H}$, and $\mathbf{H}\tilde{\mathbf{e}}_1 - \tilde{\mathbf{e}}_1\mathbf{H}$. Finally we have:

$$\begin{aligned}\mathbf{R}\frac{\partial \mathbf{R}^T}{\partial a} = \; & -\theta a\left(\frac{f'(\theta)}{\theta^2} + g'(\theta)f(\theta) - g(\theta)f'(\theta) + \frac{f(\theta)g(\theta)}{\theta}\right)\mathbf{H} \\ & -f(\theta)\tilde{\mathbf{e}}_1 \\ & \tfrac{1}{2}(f^2(\theta) + \theta^2 g^2(\theta))(\mathbf{H}\tilde{\mathbf{e}}_1 - \tilde{\mathbf{e}}_1\mathbf{H}) \end{aligned}$$

Using equation 43 the coefficient of $\mathbf{H}\tilde{\mathbf{e}}_1 - \tilde{\mathbf{e}}_1\mathbf{H}$ is $g(\theta)$. Using the expressions for f and g, the coefficient of $\mathbf{H}$ is:

$$a\frac{sin\theta - \theta}{\theta^3}$$

Finally:

$$\mathbf{R}\frac{\partial \mathbf{R}^T}{\partial a} = a\frac{sin\theta - \theta}{\theta^3}\mathbf{H} - f(\theta)\tilde{\mathbf{e}}_1 + g(\theta)(\mathbf{H}\tilde{\mathbf{e}}_1 - \tilde{\mathbf{e}}_1\mathbf{H})$$

Matrix $\mathbf{H}\tilde{\mathbf{e}}_1 - \tilde{\mathbf{e}}_1\mathbf{H}$ is given by:

$$\mathbf{H}\tilde{\mathbf{e}}_1 - \tilde{\mathbf{e}}_1\mathbf{H} = \begin{bmatrix} 0 & b & c \\ -b & 0 & 0 \\ -c & 0 & 0 \end{bmatrix}$$

If we compute the partials with respect to b and c, we get:

$$\mathbf{R}\frac{\partial \mathbf{R}^T}{\partial b} = b\frac{sin\theta - \theta}{\theta^3}\mathbf{H} - f(\theta)\tilde{\mathbf{e}}_2 + g(\theta)(\mathbf{H}\tilde{\mathbf{e}}_2 - \tilde{\mathbf{e}}_2\mathbf{H})$$

where:

$$\mathbf{H}\tilde{\mathbf{e}}_2 - \tilde{\mathbf{e}}_2\mathbf{H} = \begin{bmatrix} 0 & -a & 0 \\ a & 0 & c \\ 0 & -c & 0 \end{bmatrix}$$

and also:

$$\mathbf{R}\frac{\partial \mathbf{R}^T}{\partial c} = c\frac{sin\theta - \theta}{\theta^3}\mathbf{H} - f(\theta)\tilde{\mathbf{e}}_3 + g(\theta)(\mathbf{H}\tilde{\mathbf{e}}_3 - \tilde{\mathbf{e}}_3\mathbf{H})$$

where:

$$\mathbf{H}\tilde{\mathbf{e}}_3 - \tilde{\mathbf{e}}_3\mathbf{H} = \begin{bmatrix} 0 & 0 & -a \\ 0 & 0 & -b \\ a & b & 0 \end{bmatrix}$$

Plugging all this in equation 39 we obtain:

$$\begin{aligned} \frac{d\mathbf{ROM}}{d\mathbf{x}}\mathbf{v} &= -(\mathbf{v}.\mathbf{r}\frac{\sin\theta - \theta}{\theta^3}\mathbf{H} - f(\theta)\tilde{\mathbf{v}} + g(\theta)\widetilde{\mathbf{v}\wedge\mathbf{r}})\mathbf{ROM} \\ &= \mathbf{b}\wedge\mathbf{ROM} \end{aligned}$$

with:

$$\mathbf{b} = \frac{1 - f(\theta)}{\theta^2}(\mathbf{r}.\mathbf{v})\mathbf{r} + f(\theta)\mathbf{v} + g(\theta)\mathbf{r}\wedge\mathbf{v}$$

References

[AF85] N. AYACHE and B. FAVERJON. A fast stereo vision matcher based on prediction and recursive verification of hypotheses. In *Proceedings of the Third Workshop on Computer Vision : Representation and Control*, pages 27–37, Bellaire, USA, October 13-16 1985.

[AF86] N. AYACHE and O.D. FAUGERAS. Hyper: a new approach for the recognition and positioning of two-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):44–54, January 1986.

[AF87] N. AYACHE and O.D FAUGERAS. Building, registrating and fusing noisy visual maps. In *Proc. 1st ICCV*, June 1987.

[AHU74] V. AHO, J.E. HOPCROFT, and J.D. ULLMAN. *The Design and Analysis of Computer Algorithms.* 1974.

[AK71] Y. I. ABDEL-AZIZ and H. M. KARARA. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Symposium on Close-Range Photogrammetry*, pages 1–18, University of Illinois at Urbana Champaign, Illinois, January 26-29 1971.

[AK74] Y.I. ABDEL-AZIZ and H.M. KARARA. *Photogrammetric Potentials of Non-Metric Cameras.* Photogrammetry Series 36, University of Illinois, Urbana, Illinois, 1974. Civil Engineering Studies.

[AL87] N. AYACHE and F. LUSTMAN. Fast and reliable passive stereovision using three cameras. In *International Workshop on Industrial Applications of Machine Vision and Machine Intelligence*, Tokyo, Japan, February 1987.

[ARN86] V.I. ARNOLD. *Catastrophe Theory.* Springer-Verlag Berlin Heidelberg, 1984–1986.

[BA84] M. BRADY and H. ASADA. Smoothed local symmetries and their implementation. *Int. J. Robotics Research*, 3(3), 1984.

[BB81] H. BAKER and T.O. BINFORD. Depth from edge and intensity based stereo. In *Proceedings 7th Joint Conference on Artificial Intelligence*, pages 631–636, Vancouver, Canada, August 1981.

[BFLB87] J.D. BOISSONNAT, O.D. FAUGERAS, and E. LE BRAS. *Representing Stereo Data with the Delaunay Triangulation.* Technical Report, INRIA, Domaine de Voluceau, Rocquencourt, B.P. 105, 78153 LE CHESNAY CEDEX, 1987.

[BH86] R.C. BOLLES and P. HORAUD. 3DPO: A three-dimensional part orientation system. *Int. J. Robotics Research*, 5(3), 1986.

[BIN84] T.O. BINFORD. *Stereo Vision: Complexity and Constraints*, pages 475–487. Volume the First International Symposium of *Robotics Research*, MIT Press, Michael Brady and Richard Paul edition, 1984.

[BOI84] J.D. BOISSONNAT. Geometric structures for 3-dimensional shape representation. *ACM Transactions on Graphics*, 3(4), October 1984.

[BOI85] J.D. BOISSONNAT. Reconstruction of solids. In *First ACM Symposium on Computational Geometry*, Baltimore, June 1985.

[BOI86] J.D. BOISSONNAT. *An Automatic Solid Modeler for Robotics Applications*, pages 65–72. Volume the Third International Symposium of *Robotics Research*, MIT Press, O.D. Faugeras and G. Giralt edition, 1986.

[BOW81] A. BOWYER. Computing Dirichlet tesselations. *Computer Journal*, 24:162–166, 1981.

[BPYA85] M. BRADY, J. PONCE, A. YUILLE, and H. ASADA. *Describing Surfaces.* Technical Report, MIT AI Memo 822, January 1985.

[BRO66] D. C. BROWN. Decentering distortion of lenses. *Photogrammetric Engineering*, 32(3), May 1966.

[BRO71] D. C. BROWN. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.

[BRO84] P BROU. Finding the orientation of objects in vector maps. *Int. J. Rob. Res.*, 3(4), 1984.

[CB87] J.H. CONNEL and M. BRADY. Generating and generalizing models of visual objects. *Artificial Intelligence*, 31:159–183, 1987.

[CG87] J.P. COCQUEREZ and A. GAGALOWICZ. Mise en correspondence de régions dans une paire d'images stéréo. In *MARI 87*, Paris, mai 1987.

[CHO84] S.C. CHOU. Proving elementary geometry theorems using wu's algorithm. *Contemporary Mathematics*, 29:243, 1984.

[CRO86] J.L. CROWLEY. Representation and maintenance of a composite surface model. In *IEEE Conference on Robotics and Automation*, pages 1455–1462, San Francisco, USA, April 7-10 1986.

[DH73] O. DUDA and P.E. HART. *Pattern Classification and Scene Analysis*. Wiley-Interscience, New-York, 1973.

[DUR86] H.F. DURRANT-WHYTE. Consistent integration and propagation of disparate sensor observations. In *Proceedings 1986 IEEE Conference on Robotics and Automation*, pages 1464–1469, San Francisco, USA, April 7-10 1986.

[FAF86] O.D. FAUGERAS, N. AYACHE, and B. FAVERJON. Building visual maps by combining noisy stereo measurements. In *Proceedings 1986 IEEE Conference on Robotics and Automation*, pages 1433–1438, San Francisco, USA, April 7-10 1986.

[FAI75] W. FAIG. Calibration of close-range photogrammetric systems: mathematical formulation. *Photogrammetric Engineering and Remote Sensing*, 41(12):1479–1486, 1975.

[FGK*83] O.D. FAUGERAS, F. GERMAIN, G. KRYZE, J.D. BOISSONNAT, M. HEBERT, J. PONCE, E. PAUCHON, and N. AYACHE. *Toward a flexible Vision System*, chapter 3, pages 129–142. *Robot Vision*, IFS, UK, ed. Pugh Alan edition, 1983.

[FH86] O.D. FAUGERAS and M. HEBERT. The representation, recognition, and locating of 3d shapes from range data. *Int. J. Robotics Research*, 5(3), 1986.

[FL87] O.D. FAUGERAS and F. LUSTMAN. *Let us suppose that the world is piecewise planar*, chapter 1, pages 33–40. Volume The Third International Symposium of *Robotics Research*, MIT Press, Olivier D. Faugeras and Georges Giralt edition, 1987.

[FLT87] O.D. FAUGERAS, F. LUSTMAN, and G. TOSCANI. Motion and structure from motion from point and line matches. In *1st ICVV*, pages 25–34, 1987.

[FT86] O.D. FAUGERAS and G. TOSCANI. The calibration problem for stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR-86*, pages 15–20, Miami Beach, Florida, USA, June 22-26 1986.

[GAN77] F.R. GANTMACHER. *Matrix Theory.* Chelsea, New York, 1977.

[GAN84] S. GANAPATHY. Decomposition of transformation matrices for robot vision. In *Proceedings of Int. Conf. on Robotics and Automation*, pages 130–139, 1984.

[GDS86] E. GUREWITZ, I. DINSTEIN, and B. SARUSI. More on the benefit of a third eye for machine stereo perception. In *Proc. of the eight ICPR*, pages 966–968, Paris, FRANCE, 1986.

[GEN79] D.B. GENNERY. Stereo camera calibration. In *Proceedings Image Understanding Workshop*, pages 101–108, November 1979.

[GEN80] D.B. GENNERY. *Modeling the environment of an exploring vehicle by means of stereo vision.* Ph.D. Thesis, Stanford Artificial Intelligence Laboratory, 1980. also Artificial Intelligence Laboratory Memo 339.

[GL83] W.E.L. GRIMSON and T. LOZANO-PEREZ. Model-based recognition and localization from sparse three-dimensional data. *Int. J. Robotics Research*, 3(3):3–35, 1983.

[GPSL86] A. GERHARD, H. PLATZER, J. STEURER, and R. LENZ. Depth extraction by stereo triples and a fast correspondance estimation algorithm. In *Proc. of the 8th ICPR*, pages 512–515, Paris, FRANCE, 1986.

[GRI81] W.E.L. GRIMSON. A computer implementation of a theory of human stereo vision. *Phil. Trans. Roy. Soc. London*, B292:217–253, 1981.

[GRI87] W.E.L. GRIMSON. Recognition of object families using parametrized models. In *Proc. 1st ICCV*, pages 93–101. 1987.

[Har87] C.G. Harris. Determination of ego-motion from matched points. In *Proc. Alvey Conference*, pages 189–192, University of Cambridge, 1987.

[HIL84] E.C. HILDRETH. *The Measurement of Visual Motion.* MIT Press, Cambridge, MA, 1984.

[HOR74] B.K.P. HORN. Determining lightness from an image. *Computer Graphics and Image Processing*, 3(1):277–299, 1974.

[HOR75] B.K.P. HORN. *Obtaining Shape from Shading Information*, chapter Chapter 4 in The Psychology of Computer Vision. McGraw-Hill Book Co., New York, p.h. winston (ed.) edition, 1975. 115–155.

[HOR77] B.K.P. HORN. Image intensity understanding. *Artificial Intelligence*, 8(2):201–231, 1977.

[HOR86] B.K.P. HORN. *Robot Vision.* MIT Press, Cambridge, MA, 1986.

[HTMS82] E.L. HALL, M.B.K. TIO, C.A. McPHERSON, and F.A. SADJADI. Curved surface measurements et recognition for robot vision. In *IEEE Workshop on Industrial Applications of Machine Vision*, Conference Record, 1982.

[HUA86] T.S. HUANG. *Determining Three-Dimensional Motion and Structure from Two Perspective Views*, chapter 14. Volume in Handbook of Pattern Recognition and Image Processing, Academic Press, 1986.

[II86a] M. ITO and A. ISHII. Range and shape measurement using three-view stereo-analysis. In *Proc. CVPR86*, pages 9–14, Miami Beach, Florida, 1986.

[II86b] M. ITO and A. ISHII. Three view stereo analysis. *IEEE Trans. on PAMI*, PAMI-8(4):524–531, 1986.

[IPS85] A. IZAGUIRRE, P. PU, and J. SUMMERS. A new development in camera calibration: calibrating a pair of mobile cameras. In *Proceedings of Int. Conf. on Robotics and Automation*, pages 74–79, 1985.

[JAZ70] A.H. JAZWINSKI. *Stochastic processing and filtering theory.* Fla: Academic Press, Orlando, 1970.

[KAR79] H. M. KARARA. Handbook of non-topographic photogrammetry. *American Society of Photogrammetry*, 1979. Editor.

[KMM77] R.E. KELLY, P.R.H. McCONNEL, and S.J. MILDENBERGER. The gestalt photomapper. *Photogramm. Eng. Rem. Sens. 43*, 1407–1417, 1977.

[KOE84] J.J. KOENDERINK. *What tells us the contour about solid shape ?* Technical Report, Dept. Medical and Physiol. Physics, Univ. Utrecht, Netherlands, 1984.

[KOL74] O. KOLBL. Tangential and asymmetric lens distorsion. In *Proceedings of Symposium of Commission III*, Determined by Self Calibration, I.S.P, Stuttgart, Germany, 1974.

[LC85] J.P. LAUMOND and R. CHATILA. Position referencing and consistent world modeling for mobile robots. In *Proceedings 1985 IEEE Conference on Robotics and Automation*, pages 138–145, Saint Louis, Missouri, 1985.

[LH86a] Y. LIU and T.S. HUANG. Estimation of rigid body motion using straight line correspondences. In *Proceedings Workshop on Motion: Representation and Analysis*, pages 47–51, IEEE Computer Society, Charleston, South Carolina, USA, May 1986.

[LH86b] Y. LIU and T.S. HUANG. Estimation of rigid body motion using straight line correspondences, further results. In *Proceedings ICPR 1986*, pages 306–307, Paris, France, October 27-31 1986.

[LIN72] K. LINKWITZ. Some remarks on present investigations on calibration of close-range cameras. *International Archives of Photogrammetry*, 1972. Commission V.

[LL86] D.T. LEE and A.K. LIN. Generalized delaunay triangulation for planar graphs. *Discrete Comput. Geom. 1*, 201–217, 1986.

[LON81] H.C. LONGUET-HIGGINS. A computer algorithm for reconstructing a scene from two projection. *Nature 293*, 133–135, 1981.

[LON84] H.C. LONGUET-HIGGINS. The reconstruction of a scene from two projections - configurations that defeat the 8-point algorithm. In *Proceedings First Conference on Artificial Intelligence Applications*, pages 395–397, Denver, Colorado, USA, December 5-7 1984.

[MAL71] K. MALHOTRA. A computer program for the calibration of close-range cameras. In *Proceedings of Symposium on Close-Range Photogrammetry*, University of Illinois at Urbana Champaign, Illinois, 1971.

[MAR82] D. MARR. *Vision.* Freeman, 1982.

[MAY79] P.S. MAYBECK. Stochastic models, estimation and control. *Academic Press*, 3 volumes, 1979.

[MBK81] H.A. MARTINS, J.R. BIRK, and R.B. KELLEY. Camera models based on data from two calibration planes. *Computer Graphics and Image Processing*, 17:173–180, 1981.

[MOR80] H.P. MORAVEC. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.* Technical Report Memo 340., Stanford Artificial Intelligence Laboratory, Stanford Artificial Intelligence, 1980. Ph.D. Thesis.

[MP76] D. MARR and T. POGGIO. Cooperative computation of stereo disparity. *Science 194*, 283–287, 1976.

[MP79] D. MARR and T. POGGIO. A computational theory of stereo vision. *Proc. R. Soc. Lond. B204*, 301–328, 1979.

[MSA86a] AMAR MITICHE, STEVEN SEIDA, and J.K. AGGARWAL. Interpretation of structure and motion using straight line correspondences. In *Proceedings ICPR 1986*, pages 1110–1112, Paris, France, October 27-31 1986.

[MSA86b] AMAR MITICHE, STEVEN SEIDA, and J.K. AGGARWAL. Line based computation of structure and motion using angular invariance. In *Proceedings Workshop on Motion: Representation and Analysis*, pages 175–180, IEEE Computer Society, Charleston, South Carolina, USA, May 1986.

[MUN86] J.L. MUNDY. *Reasoning about 3-D Space with Algebraic Deduction*, pages 117–124. Volume the Third International Symposium of *Robotics Research*, MIT Press, Olivier D. Faugeras and Georges Giralt edition, 1986.

[NAG86] HANS-HELLMUT NAGEL. Image sequences - ten (octal) years- from phenomenology towards a theoretical foundation. In *Proceedings ICPR 1986*, pages 1174–1185, Paris, France, October 27-31 1986.

[NIS84] H.K. NISHIHARA. *PRISM: A practical real-time imaging stereo matcher.* Technical Report Memo 780, MIT Artificial Intelligence Lab., 1984.

[NP84] H.K. NISHIHARA and T. POGGIO. *Stereo Vision for Robotics*, pages 489–505. Volume The First International Symposium of *Robotics Research*, MIT Press, Michael Brady and Richard Paul edition, 1984.

[OK85] Y. OHTA and T. KANADE. Stereo by intra- and inter-scanline search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(2):139–154, 1985.

[OKA81] A. OKAMOTO. Orientation and construction of models. *Photogrammetric Engineering and Remote Sensing*, 47(10):1437–1454, 1981. Part I: The Orientation Problem in Close-Range Photogrammetry.

[OKA84] A. OKAMOTO. The model construction problem using the collinearity condition. *Photogrammetric Engineering and Remote Sensing*, L(6):705–711, 1984.

[OWI86] Y. OHTA, M. WATANABE, and K. IKEDA. Improving depth map by right angles trinocular stereo. In *Proc. of the eighth ICPR*, pages 519–521, Paris, FRANCE, 1986.

[PAV78] T. PAVLIDIS. *Structured Pattern Recognition.* Springer-Verlag, New York, 1978.

[PB85] J. PONCE and M. BRADY. *Toward a Surface Primal Sketch.* Technical Report MIT AI Memo 824, MIT AI, April 1985.

[PH86] M. PIETIKAINEN and D. HARWOOD. Depth from three camera stereo. In *Proc. CVPR86*, pages 2–8, Miami Beach, Florida, 1986.

[PS85] F. PREPARATA and M. SHAMOS. *Computational Geometry.* Springer-Verlag, 1985.

[PW83] E. PERVIN and J.A. WEBB. Quaternions in computer vision. In *Proc. CVPR*, Washington, 1983.

[ROD40] O. RODRIGUES. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant

de ces déplacements considérés indépendamment des causes qui peuvent les produire. *Journal De Mathématiques Pures et Appliquées*, 1st Series(5):380–440, 1840.

[SC86] R.C. SMITH and P. CHEESEMAN. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1986.

[STR84] STRAT. Recovering the camera parameters for a transformation matrix. In *Proceedings: DARPA Image Understanding Workshop*, pages 264–271, October 1984.

[SUT74] I. SUTHERLAND. Three-dimensional data input by tablet. *Proc. of le IEEE*, 62(4):453–461, 1974.

[TF87] G. TOSCANI and O.D. FAUGERAS. Structure and motion from two noisy perspective views. In *Proc. International Conference on Robotics and Automation*, Raleigh, North Carolina, USA, 1987.

[TH82] R.Y. TSAI and T.S. HUANG. Estimating three-dimensional motion parameters of a rigid planar patch, ii: singular value decomposition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-30(4), 1982.

[THO61] D'ARCY THOMPSON. *On Growth and Form.* Cambridge University Press, 1961.

[THO72] R. THOM. *Stabilité Structurelle et Morphogénèse.* W.A. Benjamin, inc., 1972.

[TSA85a] R.Y. TSAI. *Accuracy Analysis and Prediction for 3D Robotics Vision Metrology.* Technical Report, IBM Research Report RC 11348, 1985.

[TSA85b] R.Y. TSAI. *A versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology using Off-the-Shelf TV Cameras et Lenses.* Technical Report, IBM Research Report RC 11413, 1985.

[TSA86] R. Y. TSAI. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR-86*, pages 364–374, Miami Beach, Florida, USA, June 22-26 1986.

[TWK87] D. TERZOPOULOS, A. WITKIN, and M. KASS. Symmetry-seeking models for 3d object reconstruction. In *Proc. 1st ICCV*, 1987.

[VP87] A. VERRI and T. POGGIO. Against quantitative optical flow. In *1st ICCV*, 1987.

[WAT81] P.P. WATSON. Computing the n-dimensional delaunay triangulation with application to voronoi polytopes. *Computer Journal*, 24:167–172, 1981.

[WBKL84] P.H. WINSTON, T.O. BINFORD, B. KATZ, and M. LOWRY. *Learning Physical Descriptions from Functional Definitions, Examples, and Precedents*, pages 117–135. Volume the First International Symposium of *Robotics Research*, MIT Press, Brady, Michael and Paul, Richard edition, 1984.

[WON75] WONG. Mathematical formulation and digital analysis in close-range photogrammetry. *Photogrammetric Engineering and Remote Sensing*, 41(11):1355–1373, 1975.

[WU78] W.T. WU. On the decision problem and mechanization of theorem proving in elementary geometry. *Scientia Sinica 21*, 159–172, 1978.

[YAC86] M. YACHIDA. *3-D Data Acquisition by Multiple Views*, pages 11–18. Volume the Third International Symposium of *Robotics Research*, MIT Press, Cambridge, Ma., O.D. Faugeras and G. Giralt edition, 1986.

[YC78] Y. YAKIMOVSKY and R. CUNNINGHAM. A system for extracting three-dimensional measurements from a stereo pair of tv cameras. *Computer Graphics and Image Processing*, 7:195–210, 1978.

[YKK86] M. YACHIDA, Y. KITAMURA, and M. KIMACHI. Trinocular vision : new approach for correspondance problem. In *Proc. of the eighth ICPR*, pages 1041–1044, Paris, FRANCE, 1986.

[YP83a] A.L. YUILLE and T. POGGIO. *Fingerprints Theorems for Zero-Crossings.* Technical Report, MIT Artificial Intelligence Laboratory AIM-730, 1983.

[YP83b] A.L. YUILLE and T. POGGIO. *Scaling Theorems for Zero-Crossings.* Technical Report, MIT Artificial Intelligence Laboratory AIM-722, 1983.

[ZH85] X. ZHUANG and R.M. HARALICK. Two view motion analysis. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition CVPR-85*, pages 686–690, San Francisco, California, USA, June 19-23 1985.

Contents

List of Figures

List of Tables

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique